#5 26

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants:	Falko Tesch et al.		
Assignee:	Sun Microsystems, Inc.		
Title:	METHOD FOR DETERMINING RUBIES		
Serial No.:	10/054,545	Filed:	January 18, 2002
Examiner:	Unknown	Group Art Unit:	2641
Docket No.:	P-5714		

Monterey, CA
January 13, 2003

Assistant Commissioner for Patents
Washington, D.C. 20231

CLAIM FOR PRIORITY AND SUBMISSION OF PRIORITY DOCUMENT

Dear Sir:

Enclosed is a certified copy of the foreign priority application no. EP 01101380.2 for the above application.

Applicants claim the foreign priority filing date of January 22, 2001 on which the enclosed foreign priority application was filed.

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as First Class Mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, DC 20231 on January 13, 2003.

Attorney for Applicants — January 13, 2003 Date of Signature

Respectfully submitted,

Serge J. Hodgson
Attorney for Applicants
Reg. No. 40,017
(831) 655-0880

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza, Suite 220
1900 Garden Road
Monterey, CA 93940
Tel. 831-655-0880
Fax 831-655-0888

for

U.S. Serial No. 10/054,545

Eur päisches Patentamt | European Patent Office | Office eur péen des brevets

Bescheinigung | Certificate | Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

Patentanmeldung Nr. | Patent application No. | Demande de brevet n°

01101380.2

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

R C van Dijk

DEN HAAG, DEN
THE HAGUE,
LA HAYE, LE 12/12/02

Eur päisches Patentamt
European Patent Office
Office européen des brevets

Blatt 2 der Bescheinigung
Sheet 2 of the certificate
Page 2 de l'attestation

Anmeldung Nr.:
Application no.: 01101380.2
Demande n°:

Anmeldetag:
Date of filing: 22/01/01
Date de dépôt:

Anmelder:
Applicant(s):
Demandeur(s):
Sun Microsystems, Inc.
Santa Clara,@California 95054
UNITED STATES OF AMERICA

Bezeichnung der Erfindung:
Title of the invention:
Titre de l'invention:
A method for determining rubies

In Anspruch genommene Priorität(en) / Priority(ies) claimed / Priorité(s) revendiquée(s)

Staat:	Tag:	Aktenzeichen:
State:	Date:	File no.
Pays:	Date:	Numéro de dépôt:

Internationale Patentklassifikation:
International Patent classification:
Classification internationale des brevets:
G06F17/24

Am Anmeldetag benannte Vertragstaaten:
Contracting states designated at date of filing: AT/BE/CH/CY/DE/DK/ES/FI/FR/GB/GR/IE/IT/LI/LU/MC/NL/PT/SE/TR
Etats contractants désignés lors du depôt:

Bemerkungen:
Remarks:
Remarques:

AT THE TIME OF FILING OF THE APPLICATION, THE APPLICANT WAS REGISTRED AS:
Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
USA
THE REGISTRATION OF THE CHANGE HAS TAKEN EFFECT AS FROM: 27.08.2002.

A METHOD FOR DETERMINING RUBIES

FIELD OF THE INVENTION

The present invention relates to a method for determining rubies, and in particular it relates to a ruby dialogue for determining, entering and editing ruby characters.

DESECRIPTION OF THE RELATED ART

Ruby" is the commonly used name for a run of text that appears in the immediate vicinity of another run of text, referred to as the "base". Ruby serve as a pronunciation guide or a short annotation associated with the base text. Rubies are used frequently in Japan in most kinds of publications, such as books and magazines, but also in China or in Korea.

A ruby functionality is needed for computer programs which deal with Chinese, Japanese or Korean characters. Those characters - contrary to Western or Roman characters - often represent a word by a single character. Such Chinese characters (or Kanji) may be difficult to read because there are so many of them and one may not expect everybody to know all of them. Consequently, there is a need for expressing the way of reading a Chinese character by its phonetic transcription in order to enable people who do not recognize the character to nevertheless read it.

Figure 1 gives an example of a simple Japanese sentence in which the Chinese characters are superposed by their phonetic transcriptions. Those phonetic transcriptions are written using relatively simple characters which only represent the phonetic content of the Chinese character rather than its semantic meaning. Those characters which are used for the phonetic transcription are called "rubies", or in Japanese they are called "furigana".

In the example of Figure 1, the sentence would read in Roman characters "kore wa nihongo no rei" (which means "this is an example in Japanese"). Above the nouns "nihongo" and "rei" are their corresponding "rubies" in furigana. Even people who do not recognize the kanjis therefore can read the sentence because they can be assumed to recognize the relatively simple furigana transcription displayed above them. The transcription in Roman characters shown in the bottom line of Figure 1 is shown here for illustrative purposes only and will usually not be displayed.

The easy reading of rubies is possible because for the rubies only a limited set of relatively simple characters (about several dozens) are used while there exist thousands of very complicated Chinese characters (kanjis) which cannot be assumed to be known by everybody.

Additional information about rubies can be found in Appendix 1 which is a hardcopy of a webpage from the world wide web Consortium (W3C) which contains a draft specification of markup for Ruby and which also discusses some basics about ruby.

The present invention relates to the determination of which individual rubies (which ruby characters) should be displayed for a certain base text. A conventional ruby handling in this respect will now be illustrated in connection with Figure 2.

At first, a user selects a text portion for which he desires to generate rubies which then should be displayed superposed to (or above) the base text. In the example of Figure 2 it is assumed that the base text for which a ruby generation should be performed is the same as the text in Figure 1.

After having selected this text as a base text from a document, a user may then start the ruby functionality and subsequently a window as shown in Figure 2 pops up. In this window 200, one can see a table 210 which contains two columns, a first column for the selected base text and a second column for the corresponding rubies.

In a conventional ruby handling system there is provided an automatic ruby determination function for detecting individual words in the base text and for identifying corresponding ruby characters. Each individual word which is recognized as having corresponding rubies is displayed in a different row in table 210.

In table 210 one can see that the nouns "nihongo" and "rei" have been recognized by the automatic recognition functionality, and consequently each of those nouns is displayed in a different row of table 210, the noun "nihongo" in the second row and the noun "rei" in the fourth row.

The characters which are displayed in the first and the third base text row do not have corresponding ruby characters, therefore the corresponding ruby text cell is left empty. In preview window 220 the user is provided with a display of the base text together with the corresponding rubies. If the user does not agree with the proposed rubies, then he can edit the individual ruby text cells and consequently the preview also will change.

If the display in the preview window 220 is found to be correct by the user, then he may press the ok button and subsequently in the text document from which the base text has been selected there will be displayed the base text together with its corresponding rubies as it has been shown in the preview window 220.

It is an object of the present invention to provide a more flexible ruby functionality to the user than the conventional one.

SUMMARY OF THE INVENTION

According to the present invention, there is provided a method for determining rubies for generating the display of a ruby annotation to an electronic document, said method comprising: receiving a user input by a first process running on a

computer and operating on said electronic document as a working document to thereby select at least a portion of said electronic document as a base text for which a ruby annotation is to be generated; providing information on said at least one selected portion of said electronic document to a second process running on a computer, said second process enabling the display of a ruby dialogue window on a computer, said ruby dialogue window comprising a display mask comprising locations which are arranged in pairs, a first location of each pair for displaying a base text portion and a second location of each pair for displaying and editing corresponding rubies, said second process comprising: in case of an automatic ruby determination mode being set active, parsing said selected base text to detect individual words in said base text, identifying their corresponding rubies, if any, enabling the display of the detected words of said base text and enabling the display and the editing of the corresponding rubies which have been identified in said dialogue window on respectively different pairs of said first and second locations; in case of said automatic ruby determination mode being set inactive, enabling the display of each individual base text portion which has been selected on correspondingly different first locations of said pairs of locations, and enabling the user input of rubies corresponding to the base text portions displayed on said first locations at said second locations corresponding to said first locations; and if the selected text portion has been updated by a user input into said first process, receiving information on the newly selected text portion by said second process from said first process and rerunning said second process based on said updated text selection information.

The possibility to switch off the automatic ruby determination mode together with the updating of the text selection allows a very flexible handling of the ruby functionality. Rather than being dependent on the automatic recognition function a user can individually select the base text, can update the base text selection, and can then depending on his preferences insert and edit the corresponding ruby text.

According to a particular embodiment the text selection functionality comprises the capability of multi-selection for selecting multiple text portions. This enables a user

to arbitrarily select text portions in a text document which can then simultaneously be handled in a single ruby dialogue.

According to a particular embodiment the text selection functionality offers an option to search for all occurrences of a certain text string (or a certain word) in a text document. For all occurrences of such a text string or a word the ruby handling can then be carried out in the same ruby dialogue without the necessity to repeatedly carry out the ruby dialogue.

According to a particular embodiment a ruby text which has been inserted or edited once for a certain text string can be applied to all other occurrences of this text string in the text document. This facilitates an easy ruby handling for multiple occurrences of a certain base text for which the same ruby text should be generated.

According to a particular embodiment there is provided a formatting option for formatting the ruby text according to formatting templates which may be generated and selected by a user. This allows for a high flexibility of the style in which the display of the ruby characters is carried out.

BRIEF DESCRIPTION OF THE DRAWING

Figure 1 shows an example of a Japanese base text together with corresponding rubies.

Figure 2 illustrates an example of a conventional ruby handling procedure.

Figure 3 illustrates a ruby handling according to a first embodiment of the invention with automatic recognition being switched on.

Figure 4 illustrates a ruby handling according to an embodiment of the invention if automatic recognition is switched off.

Figure 5 illustrates the ruby dialogue of Figure 4 after an amendment of the base text selection.

Figure 6 illustrates the ruby dialogue window of Figure 5 after multi-selection has been used.

Figure 7 illustrates the ruby dialogue window of Figure 5 after a "search-for-all" has been performed.

Figure 8 shows a flow chart illustrating the ruby functionality according to an embodiment of the invention.

Figure 9 illustrates a multi-selection process according to an embodiment of the invention.

Figure 10 illustrates a ruby editing process in an embodiment according to the invention.

Figure 11A schematically illustrates a computer system which can be used in connection with an embodiment of the present invention.

Figure 11B illustrates a computer system which can be used in connection with the present invention.

DETAILED DESCRIPTION

A ruby dialogue window according to an embodiment of the present invention is schematically illustrated in Figure 3. The window shown in Figure 3 resembles the window as shown in Figure 2. In particular, the table 310 in window 300 has the same contents as the table 210 in the window shown in Figure 2. This is because Figure 3 shows the status where the same base text is selected in a document for a ruby functionality, moreover, in the case of Figure 3 the automatic recognition function (automatic ruby determination) is switched "on" by toggle 320.

This means that in case of Figure 3 the ruby dialogue works as the conventional ruby dialogue shown in Figure 2.

Turning now to Figure 4, there is shown the status after in the window shown in Figure 3 the toggle 320 has been pushed in order to switch the automatic ruby determination mode "off".

Rather than the automatically searching for the ruby characters corresponding to the certain base text, no automatic ruby determination (and search) is performed. Rather the selected base text 460 selected from a document 450 is displayed in the first row of the table 410. One should also note that the corresponding ruby text cell is empty since the automatic recognition function has been turned off.

Consequently the preview window only shows the base text and does not display any ruby characters.

By switching the automatic recognition function (the automatic ruby determination) "off" the whole selected base text has been moved into the first row of table 410. The base text shown in this row of table 410 corresponds to a text portion 460 which has been selected in a text document 450 schematically illustrated in Figure 4. The selection can e.g. be carried by marking the text portion 460 which is to be selected by the mouse.

The ruby dialogue according to this embodiment is so-called mode-less dialogue. This means that the ruby dialogue window 400 shown in Figure 4 while being displayed does not prevent the user from working on the text document 450 from which the text portion 460 shown in the first row of table 410 has been selected. Both windows 450 and 400 may be displayed simultaneously on a screen, and a user may just change focus between the two windows as in the conventional windows technology and may work in any of both windows which is in the focus. Contrary to a modal dialogue which restricts the user to the specific functions offered by the dialogue a modeless dialogue does not contain such limitations.

This means that a user can change the focus from the (modeless) ruby dialogue window 400 to the text document window 450 and can amend the selected text portion 460. The result of such amendment is exemplary shown in Figure 5.

In Figure 5 a text portion 560 is smaller than the text portion 460 of Figure 4 has been selected by the user in text document 550. Consequently in the first row of table 510 only a part of the base text which has been shown in table 410 of Figure 4 is displayed.

This means that due to the ruby dialogue being mode-less a user can in a very flexible manner select a base text for which a ruby functionality is desired. The selected text can then easily be altered due to the mode-less design of the ruby dialogue window. Despite the ruby dialogue being open a user can still change focus to working document 500, perform normal operations in the working document (such as text editing) or may even change the selection of the base text. Such an amendment of the base text selection will then directly affect the ruby dialogue the content of which will be updated based on the newly selected text.

Working document 500 may be any document on which some operation is performed, such as a text document edited by a word processor, a website

displayed by a browser, an image containing some text, or any other document containing some text which can be selected as base text for the ruby generation.

After having selected the base text as shown in Figure 5, a user may according to his preferences enter the corresponding ruby text into table 510. If the user is satisfied with the selected base text and the ruby text which he has inputted he may push button 540 to apply the ruby functionality to the text document 550. Then for portion 560 of text document 550 the ruby characters as may have been shown in preview window 530 will be displayed.

Figure 6 illustrates the ruby dialogue according to an embodiment of the present invention in connection with the so-called multi-selection capability. The multi-selection capability is a feature which is well known and which is offered by the Star Office Software produced by Sun Microsystems. By pressing the Ctrl-key on the keyboard a user can using the left mouse button select multiple text portions in a text document. This is exemplarily shown in connection with the ruby dialogue in Figure 6.

In text document 600 two text portions 660 and 665 have been selected using the multi-selection functionality. The first selected portion 660 is displayed as base text in the first row in table 610, the second selected text portion 665 is displayed in the second row of table 610. For the two text portions selected as the base text a user can then insert and edit a corresponding ruby text into table 610. This has the advantage that by carrying out the ruby dialogue only once a user can nevertheless apply the ruby functionality for arbitrarily selected text portions of the text document 600. Preview window 640 shows a preview of the selected base text and the corresponding rubies. In case of Figure 6 no rubies have been inputted yet, consequently no rubies are displayed.

A particular feature of the multi-selection functionality is the so-called "search for all" option which is also known from the StarOffice Software produced by Sun Microsystems. Using the "search" option in the task bar and selecting the option

"search for all" a user can select for all occurrences of a certain word or a certain text in a text document.

The application of this feature in connection with the ruby dialogue is illustrated now in connection with Figure 7. In tables 710 of the ruby dialogue window 700 each row contains the same base text, namely the term "nihongo". This is a result of searching for all occurrences of the term "nihongo" in text document 750. A user may now enter a corresponding ruby text into the first row of table 710, he may then press the "apply to all" button and consequently the ruby text inserted into the first row of table 710 will be pasted into all other rows. Thereby a user can easily generate an arbitrary ruby text for each occurrence of a certain word or a certain text string in a text document 750.

Figure 8 shows a flowchart illustrating the ruby functionality according to an embodiment of the present invention.

The left-hand-side of Figure 8 illustrates a first process running on a computer, the right-hand side illustrates a second process running on a computer. The first process may for example correspond to a part of an application program by means of which some operations on a working document are performed. The second process may e.g. correspond to operations performed by a ruby dialogue (or a ruby dialogue window). In a particular embodiment the process 1 may be a parent process (or a parent window) and process 2 may be a child process (or child window) which is called from parent process 1. For example, child process 2 may be called from parent process 1 by pressing a certain key of the keyboard or by selecting a menu item from the task bar which may e.g. be labeled "ruby functionality".

In step 810 the first process receives a user input from a user. Thereby a text portion of the working document 500 of Fig. 5 is selected. This can e.g. be done by using the mouse for marking some text portion in the well known manner. In step 820 some information on the selected text is provided to the second process. This

can be done in many ways. E.g. the selected text may directly be copied into the second process, or a pointer to a starting location of the selected text together with a length of the selected text could be passed over from the first to the second process. Another possibility would be to deliver a stream from process 1 to process 2, with an EOF marker indicating the end of the selected text.

In step 840 belonging to process 2 the information on the selected text is received. In step 845 it is then determined whether the automatic ruby detection mode is set active. If yes, then in steps 850 to 865 a conventional ruby handling is performed. At first the selected text is parsed to detect individual words by any known parsing method, then corresponding rubies are identified by looking in a database whether for the detected words the exist any rubies. The rubies so identified are then displayed together with the corresponding base text as shown e.g. in Fig. 3

If in step 845 the automatic determining mode is detected to be inactive, then in step 875 all the selected text portion is displayed in the first row of table 510 as shown in Fig. 5. Since no automatic detection is carried out no individual words are detected, consequently in step 875 the whole selected text is displayed in a single row of table 510 and no rubies are displayed. Table 510 is a mask for displaying the base text and the corresponding rubies. Each row of table 510 comprises two cells, the left cell for the base text and the right cell for the corresponding rubies. If a single base text portion has been selected and the corresponding selection information has been received by process 2, and if further the automatic determination mode is set inactive, then the whole base text portion is displayed in the left cell of the first row of table 510. The right cell is provided for allowing a user to input the rubies corresponding to the selected base text.

In step 880 a user may then input the rubies corresponding to the selected base text into the second cell of the first row of table 510.

If a user is not satisfied with the selected base text, then at any time he may amend the text selection in process 1. This can be done because process 1 and process 2

do not block each other but rather are running in parallel to each other. E.g. by changing the focus window as in the conventional windows system a user may switch focus between process 1 (which may be a word processing operation) and process 2 (the ruby dialogue window). After having changed focus to process 1 or to its corresponding window (which is a well known operation and is therefore not shown in Fig. 8) a user may request to alter the text selection (e.g. by pressing the left mouse button and marking some text). This operation is detected in step 830, the corresponding user input is received in step 810 and passed over to process 2 in step 820. Such an alteration of the selected text portion will be recognized by step 885 (or step 870 in case of automatic recognition) of process 2 and will cause process 2 to be reexecuted.

Consequently a continous updating of the text selection information is carried out, and an alteration of the base text selection is possible at any time (the ruby dialogue can be said to be modeless). This provides a very flexible ruby dialogue to the user.

According to a further embodiment in case of automatic recognition being set active after step 870 and before process 2 ends there may be carried out a further check (not shown in Fig. 8) whether the automatic recognition is still active. If no, the procedure would then jump to step 875 and the operation will proceed as described before. Since no automatic recognition would be performed, the column for the rubies (right column in table 510 of Fig. 5) would then become empty as soon as the automatic recognition function would have been switched of, furthermore the whole selected base text would move into the first row of table 510 (because no words are detected). If the automatic recognition would still be active, then process 2 would end.

This has the advantage that in case of the automatic recognition being inactive the granularity of the base text portions and how they are displayed in table 510 only depends on the user selection of the base text and not on the results of the automatic recognition function (as e.g. in case of Fig. 3).

Figure 9 shows a flowchart illustrating the text selection of multiple text portions in connection with its display in the ruby table of the ruby dialogue window. As has been explained before already, multiple text portions can be selected as base text. In step 900 a first text portion is selected, this first portion is displayed in the first base text row as illustrated in step 910. If additional text is selected in step 920, then this text is displayed in the next base text row of the table in the ruby dialogue window according to step 930 and the procedure returns to step 920 due to the mode-less character of the ruby dialogue. If no additional text is selected, then the procedure ends.

As explained before, when text is selected corresponding information is passed over from process 1 to process 2. If multiple text portions are selected, then a corresponding plurality of informations is passed over, e.g. a plurality of pointers of the like. The ruby dialogue window is generated by process 2 and it comprises a mask in which locations for display and editing are arranged in pairs, each pair comprising a first location for a base text portion and a second location for corresponding rubies (if any). If multiple text portions are passed over to process 2, then they are displayed in correspondingly different first locations. If the mask takes the form of a table, as in Figures 1 to 7, then each individual selected base text portion is displayed in a different row in the left cell of said row.

Figure 10 shows a flowchart illustrating the "search for all" option in connection with the ruby dialogue. A ruby text in the first row of the table in the ruby dialogue window is edited in step 100. Check box 110 checks whether or not other rows are to be edited (step 120 if yes), if no then check box 130 asks whether or not the first ruby text row should be applied to all other ruby text rows. If yes, then in step 140 the text of the first ruby text row is pasted into all other ruby text rows, the result may again be amended by the user by returning to step 120 and then the procedure ends. Thereby a certain ruby text can be easily generated for multiple occurrences of a certain base text in a text document.

For a ruby text generated as described before a user may select a certain style or format according to his preferences. This will now be explained by referring again to Figure 3. Figure 3 shows a pull-down menu 350 from which a user can select a certain template. The concept of templates is known from the StarOffice Software produced by Sun Microsystems. It offers the user the possibility to define and select certain formats which then are applied to a text. A template may e.g. consist in the selection of a certain font, a certain font size, whether or not the style is bold-faced or not, courier or not, etc. Such templates may be generated using the stylist-option 360 shown in Figure 3. This option is known from the StarOffice Software already and will therefore not be explained in detail here.

After the base text and the corresponding rubies have been determined as described before (and accepted by the user, e.g. by pressing an OK button not shown), they may be displayed in the working document. The display of the rubies can be carried out in a conventional manner and will therefore not be described in detail here.

The present invention is applicable to a hardware configuration like a personal computer or a work station as illustrated schematically in Figure 11A. The computer may comprise a central processing unit CPU 26, an input output I/O unit 21, an internal memory 22 and an external memory 24. The computer may further comprise standard input devices like a keyboard 23, a mouse 28 or a speech processing means (not illustrated).

The invention, however, may also be applied to a client-server configuration as illustrated in Figure 11B. The document may be displayed on a display screen of a client device 60 while some or all steps of the method as illustrated before in connection with Figures 1 to 7 are carried out on one or more server computer accessible by a client device over a data network such as the internet using a browser application or the like.

While the invention has been particularly shown with the reference to embodiments thereof, it will be understood by those skilled in the art that various other changes in the form and details may be made therein without departing from the spirit and scope of the invention. For example, the arrangement of a display of selected base text and corresponding ruby not necessarily has to take place in the form of a table, any other arrangement can be chosen as well which allows the user to identify pairs of base text and corresponding ruby, e.g. an arrangement in pairs of lines one after another.

CLAIMS

1. A method for determining rubies for generating the display of a ruby annotation to an electronic document, said method comprising:

receiving a user input by a first process running on a computer and operating on said electronic document as a working document to thereby select at least a portion of said electronic document as a base text for which a ruby annotation is to be generated;

providing information on said at least one selected portion of said electronic document to a second process running on a computer, said second process enabling the display of a ruby dialogue window on a computer, said ruby dialogue window comprising a display mask comprising locations which are arranged in pairs, a first location of each pair for displaying a base text portion and a second location of each pair for displaying and editing corresponding rubies, said second process comprising:

in case of an automatic ruby determination mode being set active, parsing said selected base text to detect individual words in said base text, identifying their corresponding rubies, if any, enabling the display of the detected words of said base text and enabling the display and the editing of the corresponding rubies which have been identified in said dialogue window on respectively different pairs of said first and second locations;

in case of said automatic ruby determination mode being set inactive, enabling the display of each individual base text portion which has been selected on correspondingly different first locations of said pairs of locations, and enabling the user input of rubies corresponding to the base text portions displayed on said first locations at said second locations corresponding to said first locations; and

if the selected text portion has been updated by a user input into said first process, receiving information on the newly selected text portion by said second process from said first process and rerunning said second process based on said updated text selection information.

2. A method for determining rubies for generating a ruby annotation to an electronic document, said method comprising:

selecting a base text for which a ruby annotation is to be generated by a user input into a first process running on a computer and operating on said electronic document as a working document to thereby select at least a portion of said electronic document;

after information on said at least one selected portion of said electronic document having been provided to a second process running on a computer, displaying by said second process a ruby dialogue window on a computer, said ruby dialogue window comprising a display mask comprising locations which are arranged in pairs, a first location of each pair for displaying a base text portion and a second location of each pair for displaying and editing corresponding rubies, said second process comprising:

in case of an automatic ruby determination mode being set active, after having parsed said selected base text to detect individual words in said base text, and after having identified their corresponding rubies, if any, displaying the detected words of said base text and displaying and editing the corresponding rubies which have been identified in said dialogue window on respectively different pairs of said first and second locations;

in case of said automatic ruby determination mode being set inactive, displaying of each individual base text portion which has been selected on correspondingly different first locations of said pairs of locations, and inputting by a user rubies corresponding to the base text portions displayed on said first locations at said second locations corresponding to said first locations; and

if the selected text portion has been updated by a user input into said first process, receiving information on the newly selected text portion by said second process from said first process and rerunning said second process based on said updated text selection information.

3. The method of claim 1 or 2, wherein

multiple text portions can be selected by said user input.

4. The method of one of claims 1 to 3, wherein

in case of said automatic ruby determination being set active, each individual word which has been detected in a selected base text is displayed in a different row of a first column of a table, with the corresponding ruby characters being displayed in the same row but in the second column of said table.

5. The method of one of claims 1 to 4, wherein

in response to said user input all occurrences of a certain word or text string in a document are selected as selected text portions.

6. The method of one of claims 1 to 5, further comprising:

in case of all occurrences of a certain word or text string being selected as a base text for ruby generation, determining the corresponding ruby text by inserting said corresponding ruby text only once for all occurrences of said text string selected as base text.

7. The method of one of claims 1 to 6, further comprising:

providing a formatting option for formatting the ruby text according to formatting templates which may be generated and selected by a user.

8. The method of one of claims 1 to 7, wherein

said pair of first and second locations are the fields of a row of a table comprising two columns; and

said ruby dialogue window is a modeless dialogue window.

9. A software tool for determining rubies for generating the display of a ruby annotation to an electronic document, said software tool comprising:

a module for receiving a user input by a first process running on a computer and operating on said electronic document as a working document to thereby select at least a portion of said electronic document as a base text for which a ruby annotation is to be generated;

a module for providing information on said at least one selected portion of said electronic document to a second process running on a computer, said second process enabling the display of a ruby dialogue window on a computer, said ruby dialogue window comprising a display mask comprising locations which are arranged in pairs, a first location of each pair for displaying a base text portion and a second location of each pair for displaying and editing corresponding rubies, said second process comprising:

a module for in case of an automatic ruby determination mode being set active, parsing said selected base text to detect individual words in said base text, identifying their corresponding rubies, if any, enabling the display of the detected words of said base text and enabling the display and the editing of the corresponding rubies which have been identified in said dialogue window on respectively different pairs of said first and second locations;

a module for in case of said automatic ruby determination mode being set inactive, enabling the display of each individual base text portion which has been selected on correspondingly different first locations of said pairs of locations, and enabling the user input of rubies corresponding to the base text portions displayed on said first locations at said second locations corresponding to said first locations; and

a module for if the selected text portion has been updated by a user input into said first process, receiving information on the newly selected text portion by said second process from said first process and rerunning said second process based on said updated text selection information.

10. A software tool for determining rubies for generating a ruby annotation to an electronic document, said software tool comprising:

a module for selecting a base text for which a ruby annotation is to be generated by a user input into a first process running on a computer and operating on said electronic document as a working document to thereby select at least a portion of said electronic document;

a module for after information on said at least one selected portion of said electronic document having been provided to a second process running on a

computer, displaying by said second process a ruby dialogue window on a computer, said ruby dialogue window comprising a display mask comprising locations which are arranged in pairs, a first location of each pair for displaying a base text portion and a second location of each pair for displaying and editing corresponding rubies, said second process comprising:

a module for in case of an automatic ruby determination mode being set active, after having parsed said selected base text to detect individual words in said base text, and after having identified their corresponding rubies, if any, displaying the detected words of said base text and displaying and editing the corresponding rubies which have been identified in said dialogue window on respectively different pairs of said first and second locations;

a module for in case of said automatic ruby determination mode being set inactive, displaying of each individual base text portion which has been selected on correspondingly different first locations of said pairs of locations, and inputting by a user rubies corresponding to the base text portions displayed on said first locations at said second locations corresponding to said first locations; and

a module for if the selected text portion has been updated by a user input into said first process, receiving information on the newly selected text portion by said second process from said first process and rerunning said second process based on said updated text selection information.

11. The software tool of claim 9 or 10, wherein

multiple text portions can be selected by said user input.

12. The software tool of one of claims 9 to 11, wherein

in case of said automatic ruby determination being set active, each individual word which has been detected in a selected base text is displayed in a different row of a first column of a table, with the corresponding ruby characters being displayed in the same row but in the second column of said table.

13. The software tool of one of claims 9 to 12, wherein

in response to said user input all occurrences of a certain word or text string in a document are selected as selected text portions.

14. The software tool of one of claims 9 to 13, further comprising:

in case of all occurrences of a certain word or text string being selected as a base text for ruby generation, determining the corresponding ruby text by inserting said corresponding ruby text only once for all occurrences of said text string selected as base text.

15. The software tool of one of claims 9 to 14, further comprising:

providing a formatting option for formatting the ruby text according to formatting templates which may be generated and selected by a user.

16. The software tool of one of claims 9 to 15, wherein

said pair of first and second locations are the fields of a row of a table comprising two columns; and

said ruby dialogue window is a modeless dialogue window.

17. A computer program product comprising computer executable program code for enabling a computer to carry out a method according to one of claims 1 to 8.

18. A computer program comprising computer executable program code for enabling a computer to carry out a method according to one of claims 1 to 8.

ABSTRACT

A method for determining rubies for generating the display of a ruby annotation to an electronic document, said method comprising: receiving a user input by a first process running on a computer and operating on said electronic document as a working document to thereby select at least a portion of said electronic document as a base text for which a ruby annotation is to be generated; providing information on said at least one selected portion of said electronic document to a second process running on a computer, said second process enabling the display of a ruby dialogue window on a computer, said ruby dialogue window comprising a display mask comprising locations which are arranged in pairs, a first location of each pair for displaying a base text portion and a second location of each pair for displaying and editing corresponding rubies, said second process comprising: in case of an automatic ruby determination mode being set active, parsing said selected base text to detect individual words in said base text, identifying their corresponding rubies, if any, enabling the display of the detected words of said base text and enabling the display and the editing of the corresponding rubies which have been identified in said dialogue window on respectively different pairs of said first and second locations; in case of said automatic ruby determination mode being set inactive, enabling the display of each individual base text portion which has been selected on correspondingly different first locations of said pairs of locations, and enabling the user input of rubies corresponding to the base text portions displayed on said first locations at said second locations corresponding to said first locations; and if the selected text portion has been updated by a user input into said first process, receiving information on the newly selected text portion by said second process from said first process and rerunning said second process based on said updated text selection information.

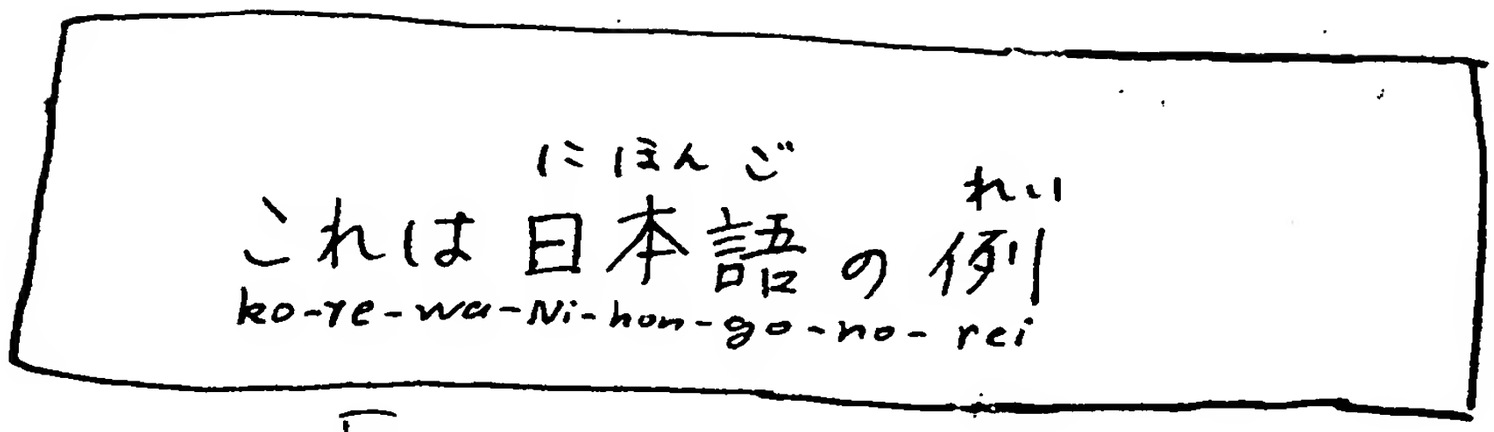

にほんご
これは日本語の例
れい
ko-re-wa-Ni-hon-go-no-rei

Fig. 1

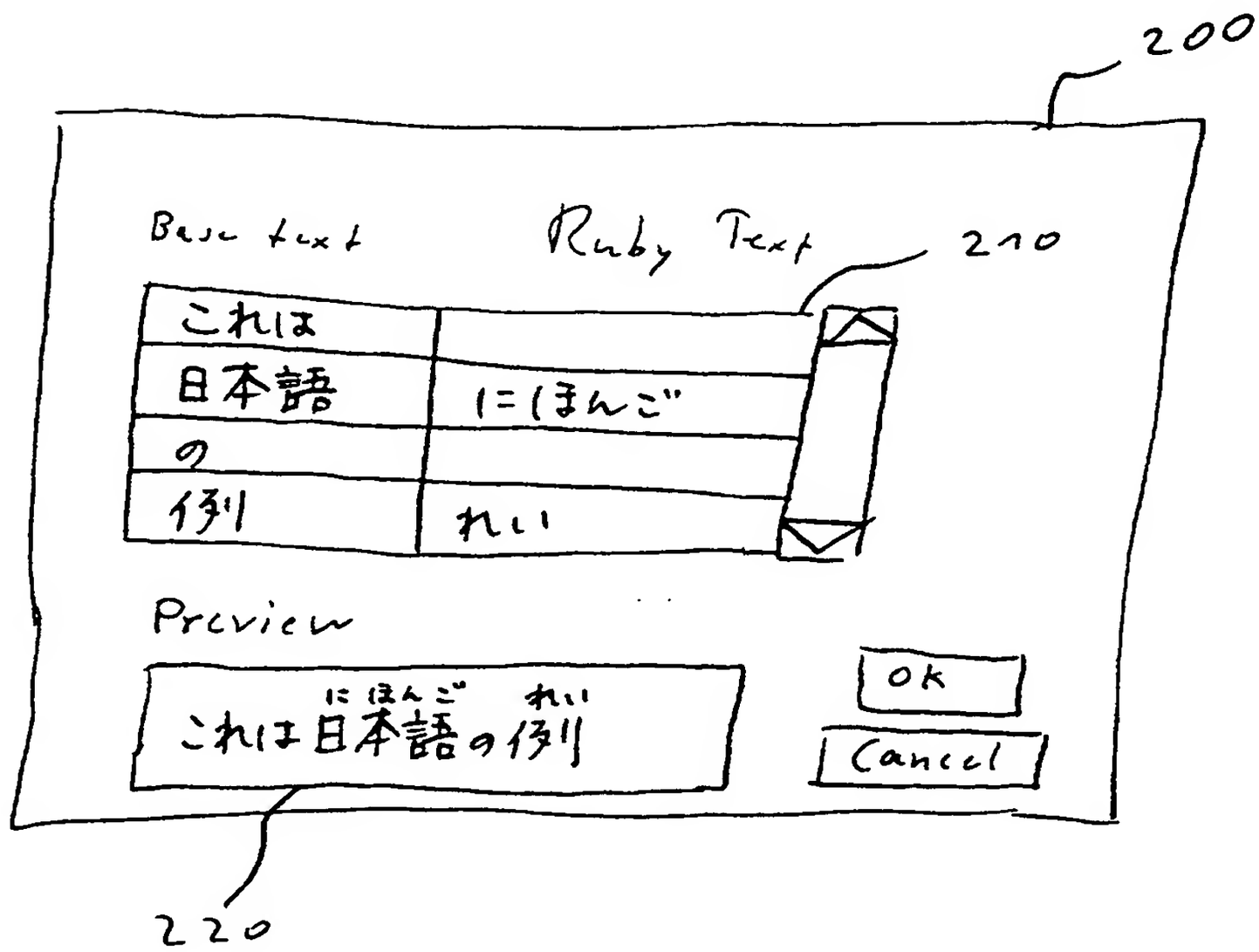

200
Base text
Ruby Text
210
これは
日本語
にほんご
の
例
れい
Preview
にほんご れい
これは日本語の例
OK
Cancel
220

Fig. 2

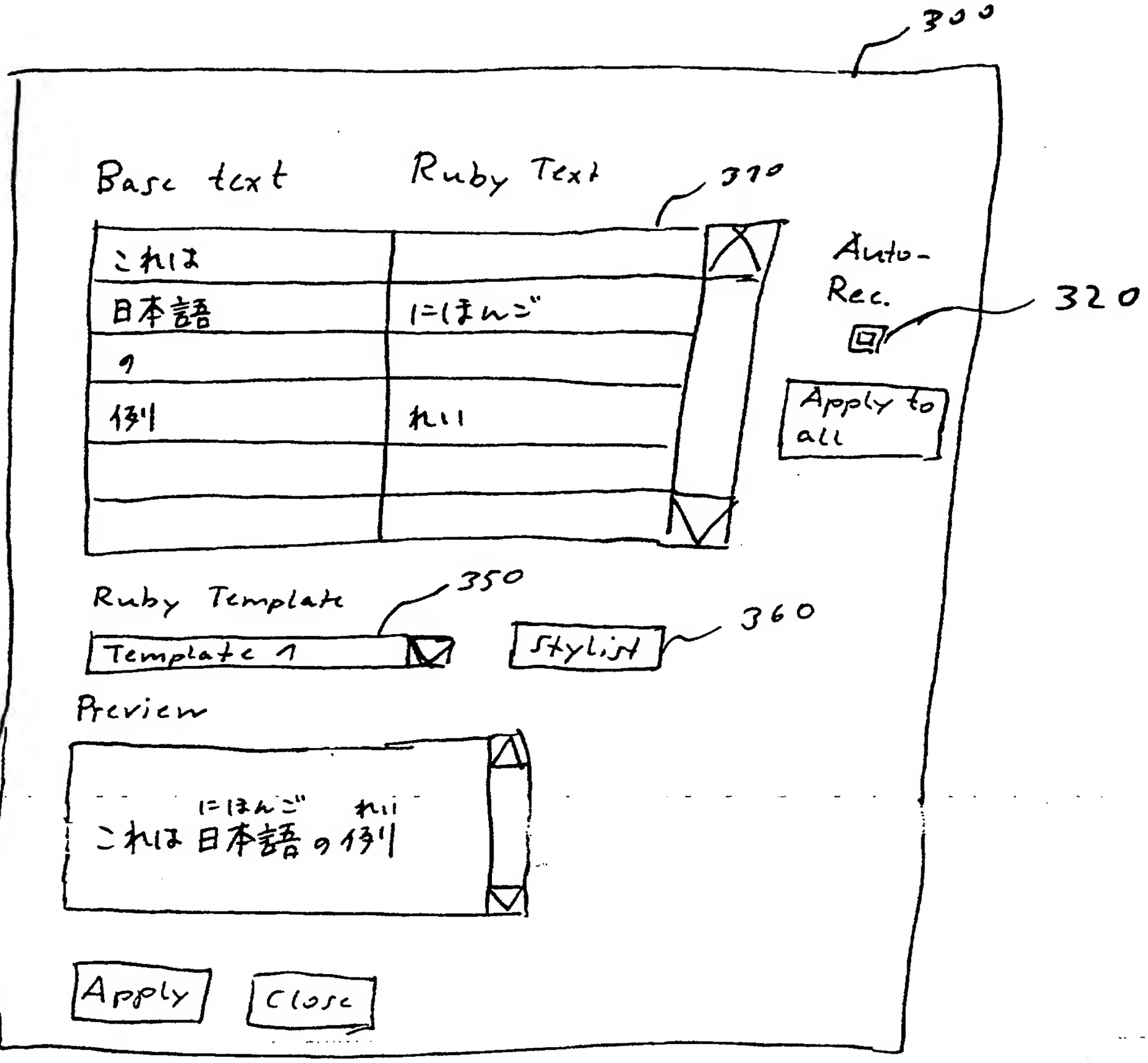

300
Base text
Ruby Text
310
これは
日本語
にほんご
の
例
れい
Auto-
Rec.
320
Apply to
all
Ruby Template
350
Template 1
Stylist
360
Preview
にほんご れい
これは 日本語 の 例
Apply
Close

Fig. 3

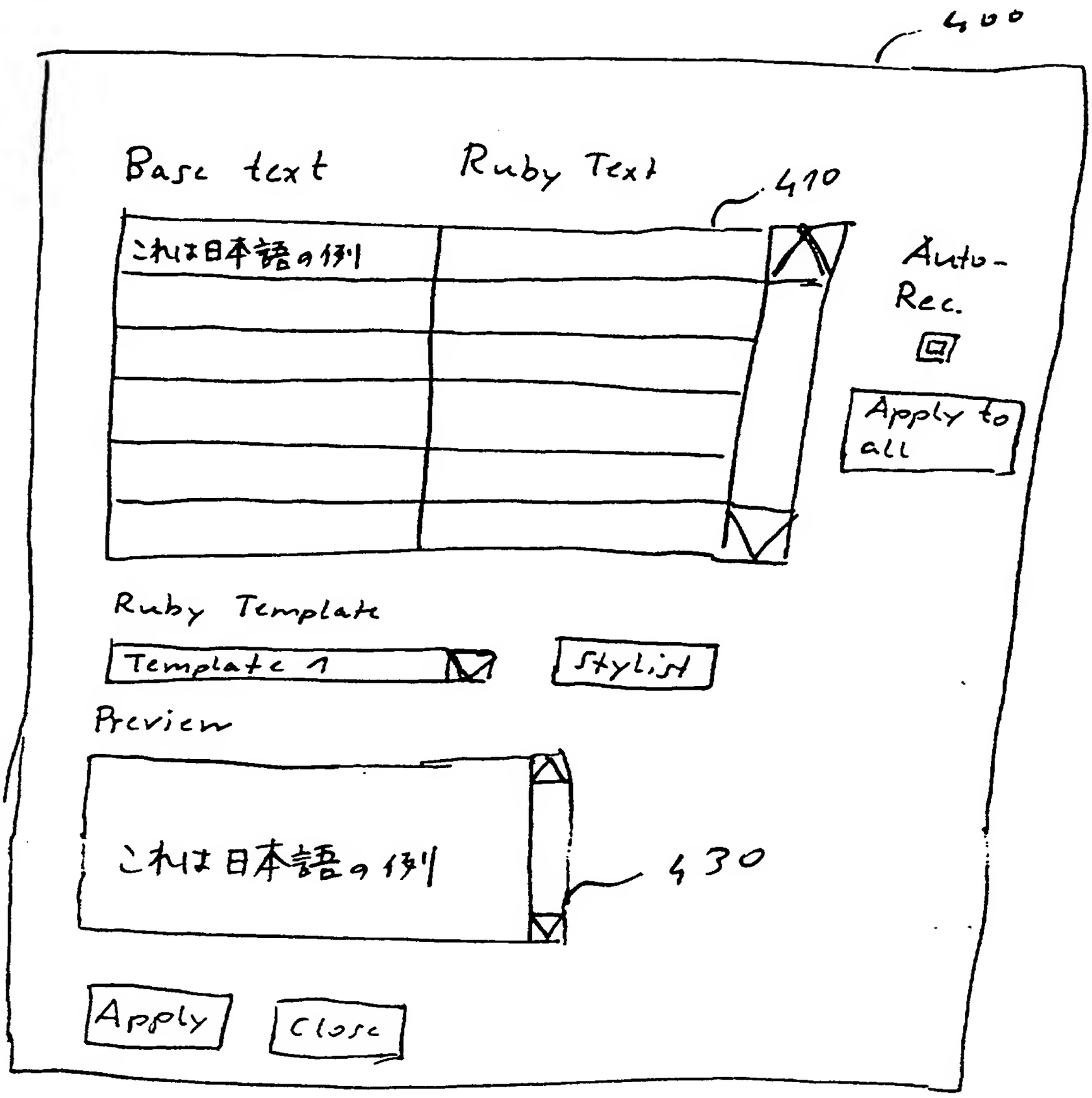
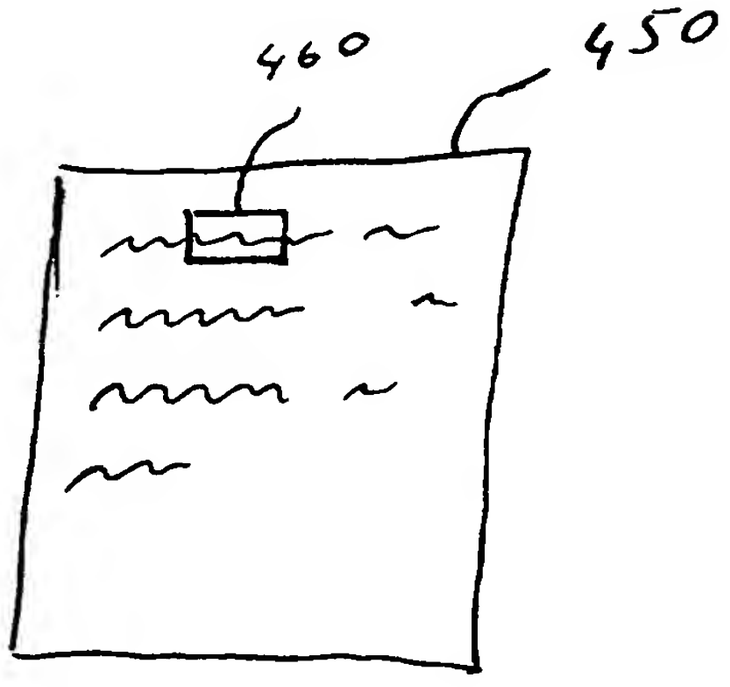

400
Base text
Ruby Text
410
これは日本語の例
Auto-Rec.
Apply to all
Ruby Template
Template 1
Stylist
Preview
これは日本語の例
430
Apply
Close

Fig. 4


460
450

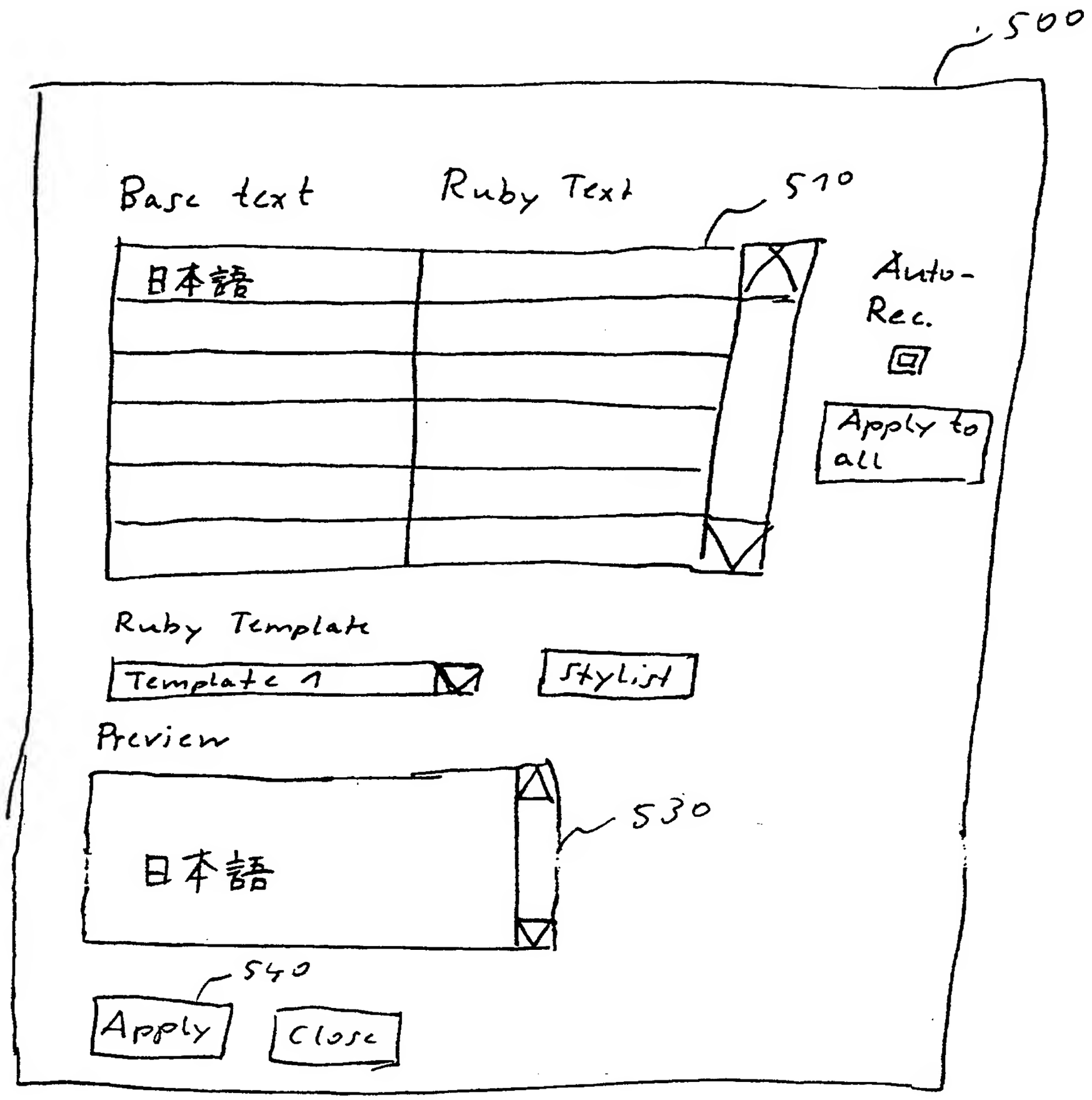
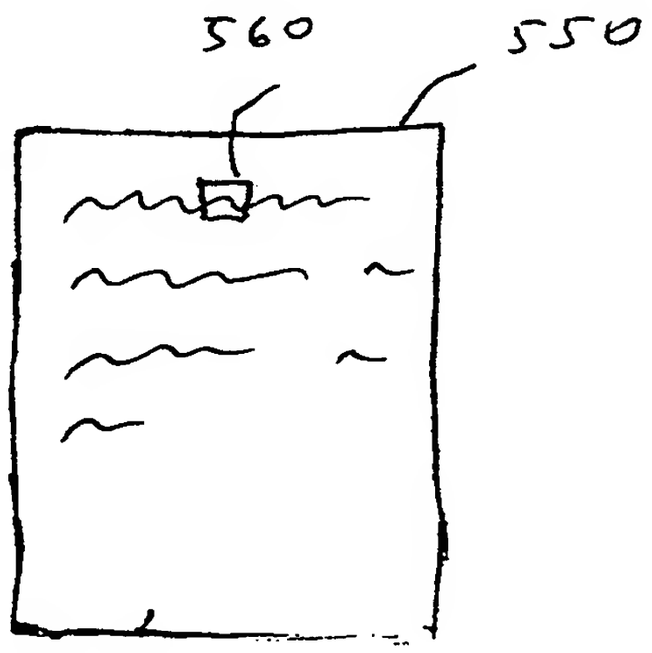

500
Base text
Ruby Text
510
日本語
Auto-
Rec.
Apply to
all
Ruby Template
Template 1
Stylist
Preview
日本語
530
540
Apply
Close

Fig. 5


560
550

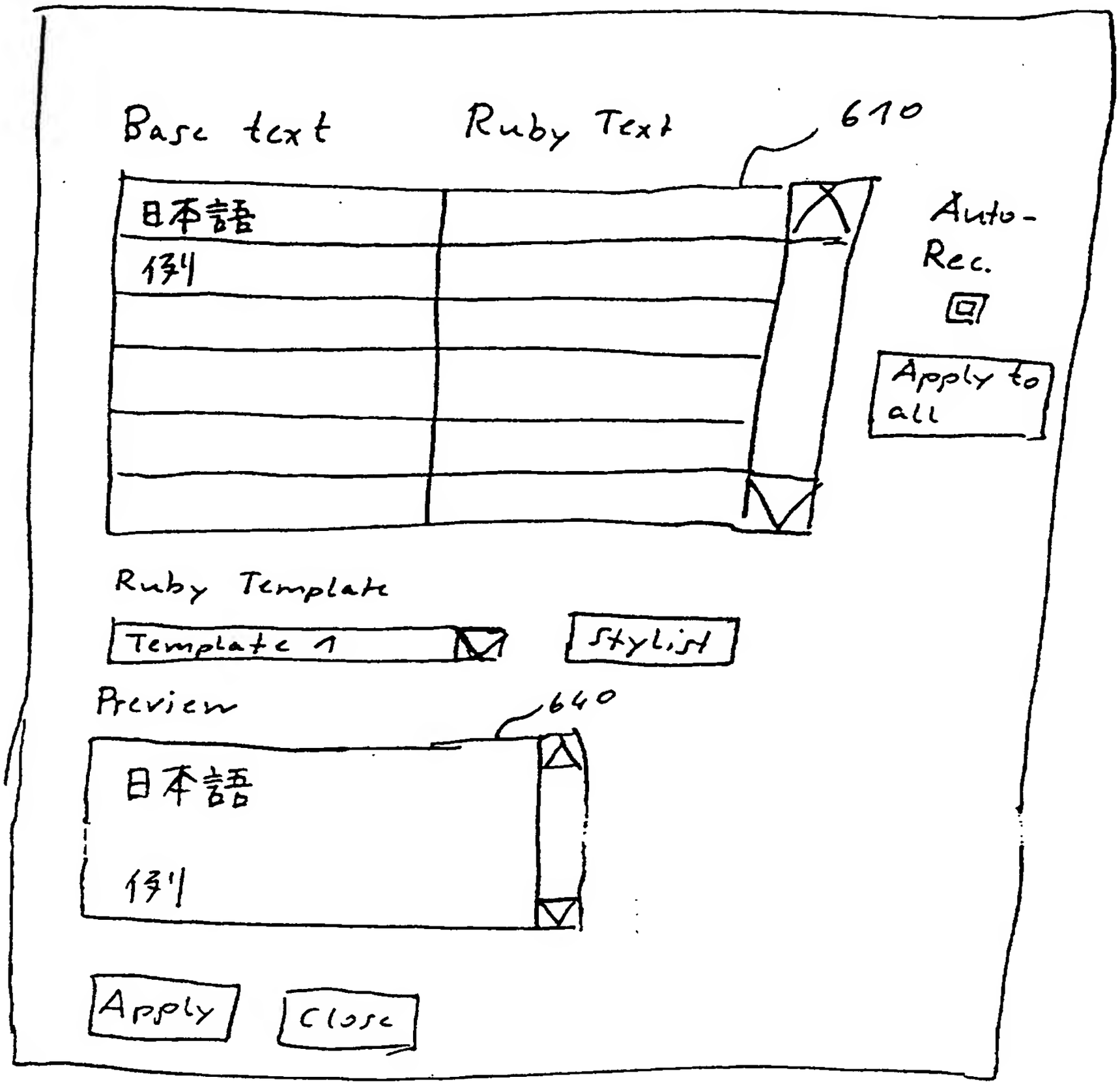
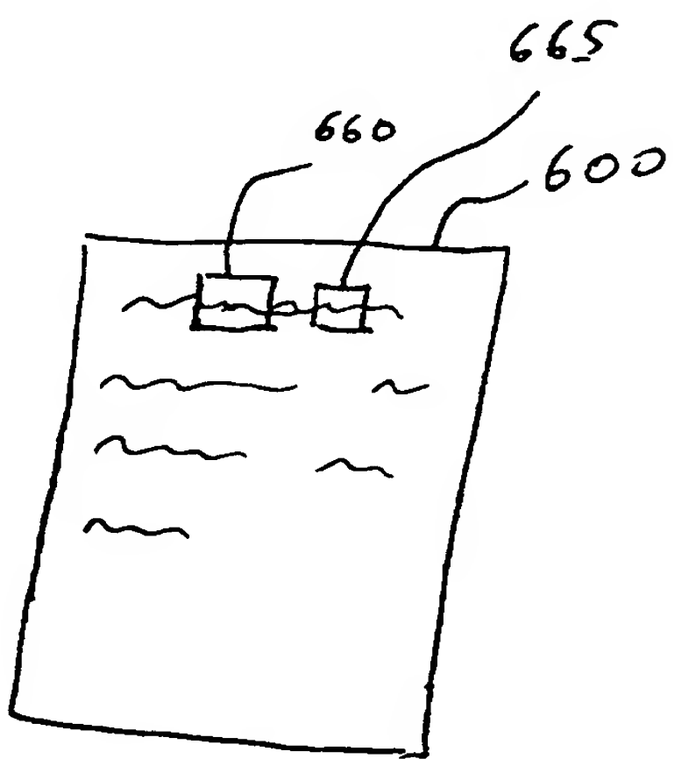

Base text
Ruby Text
610
日本語
例
Auto-
Rec.
Apply to
all
Ruby Template
Template 1
Stylist
Preview
640
日本語
例
Apply
Close

Fig. 6


665
660
600

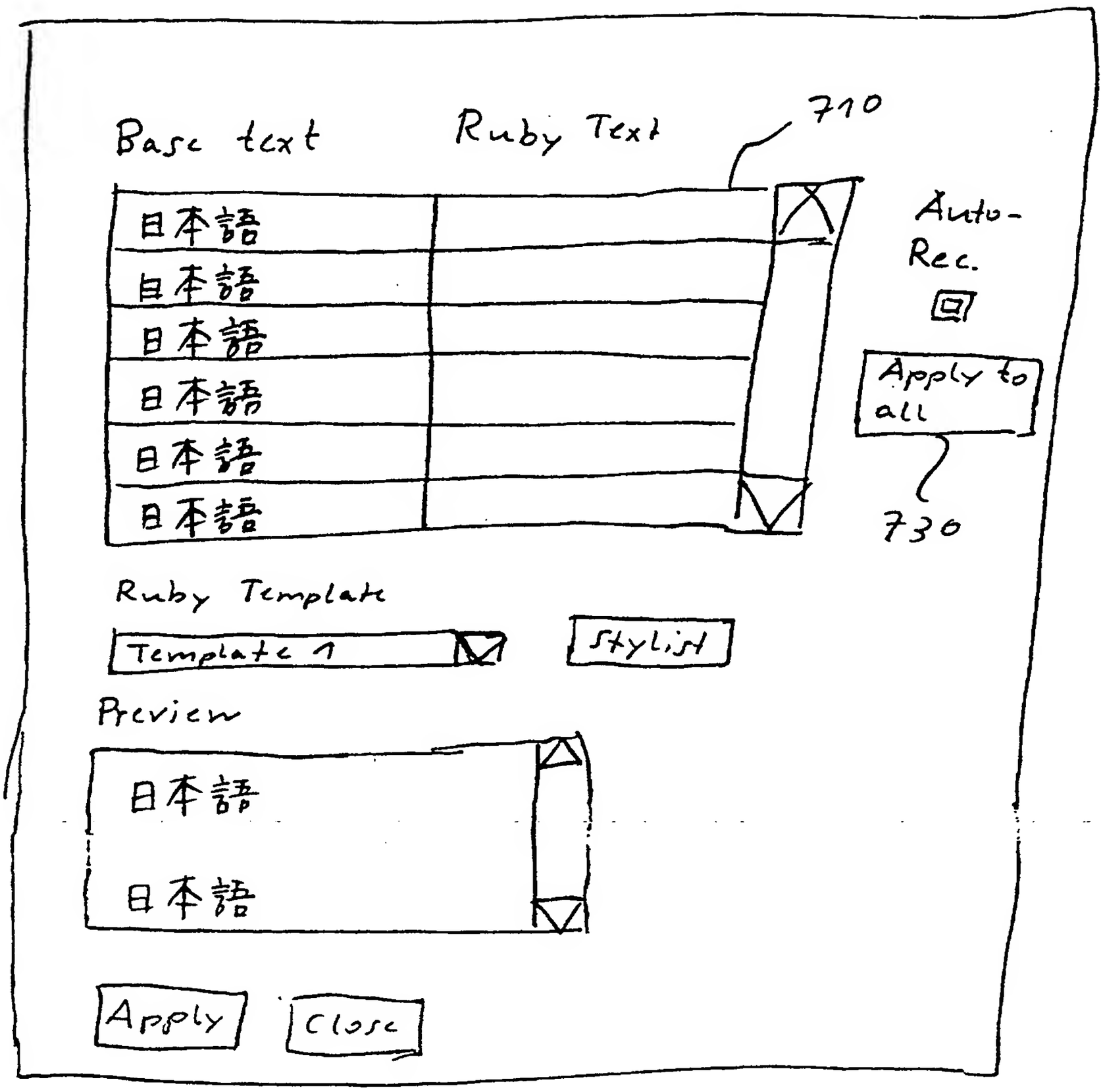
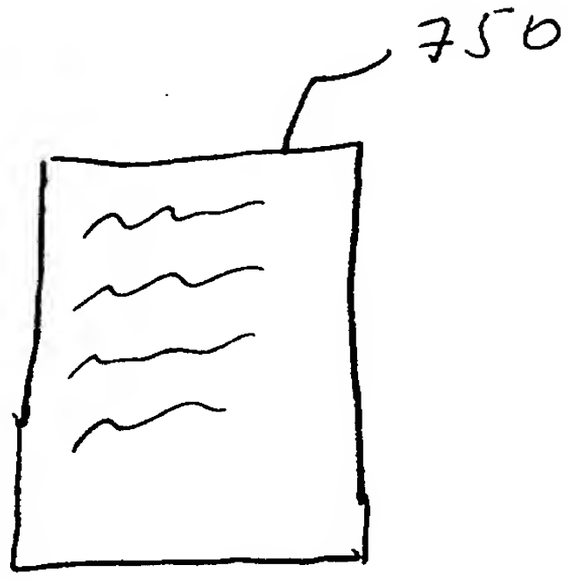
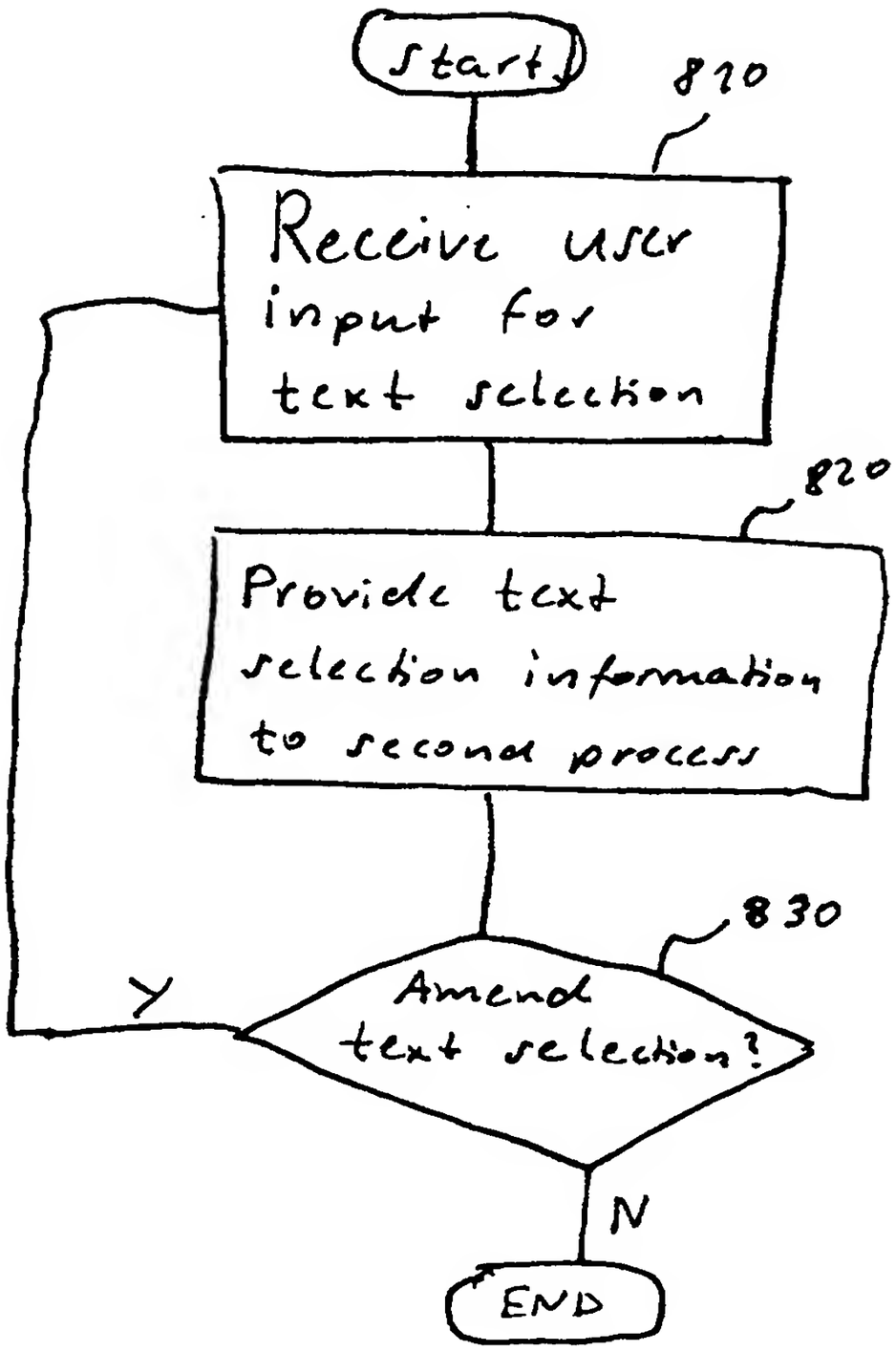
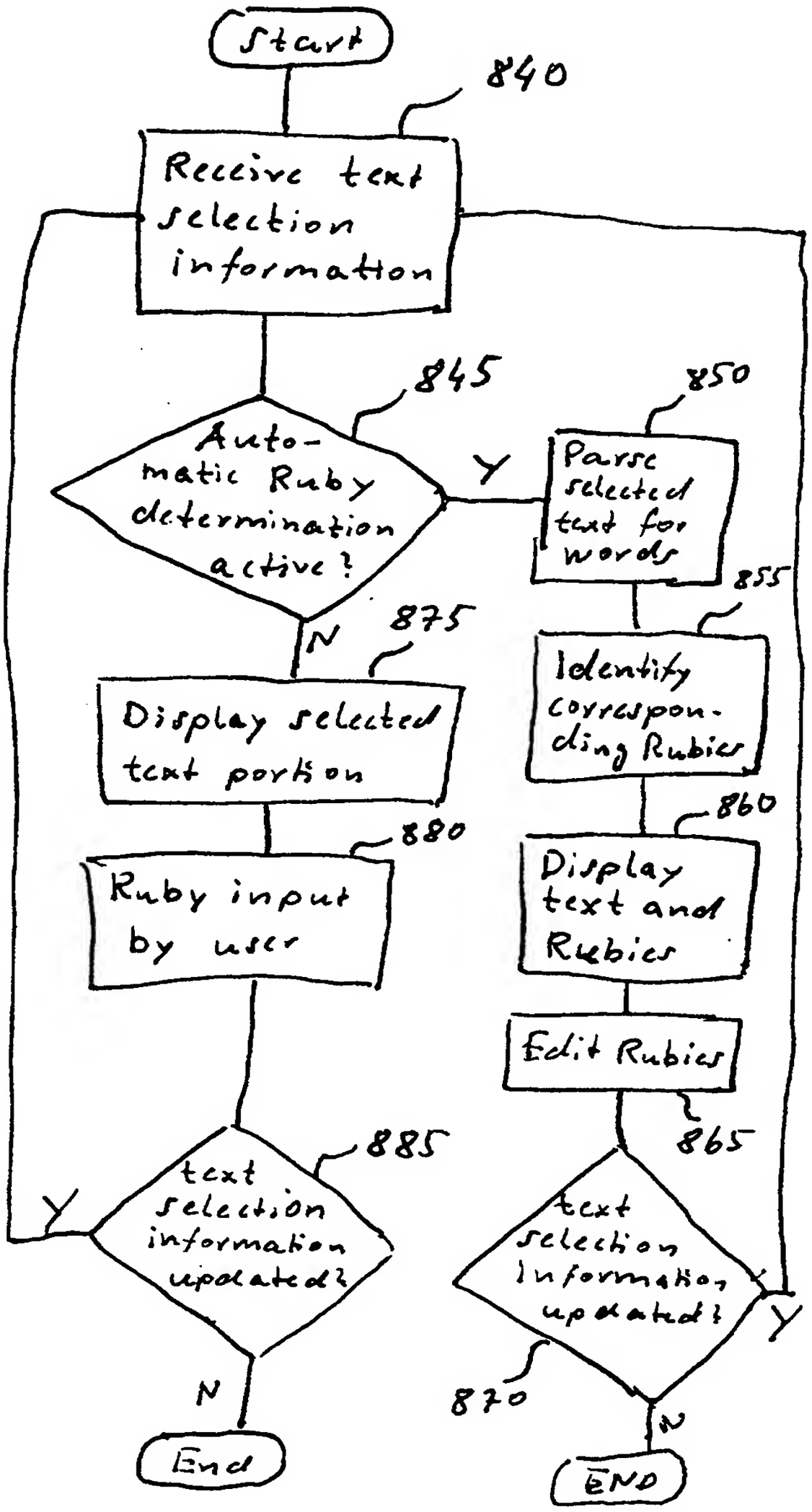

Base text
Ruby Text
710
日本語
日本語
日本語
日本語
日本語
日本語
Auto-
Rec.
Apply to all
730
Ruby Template
Template 1
Stylist
Preview
日本語
日本語
Apply
Close

Fig. 7


750



First Process



Second Process

Fig. 8

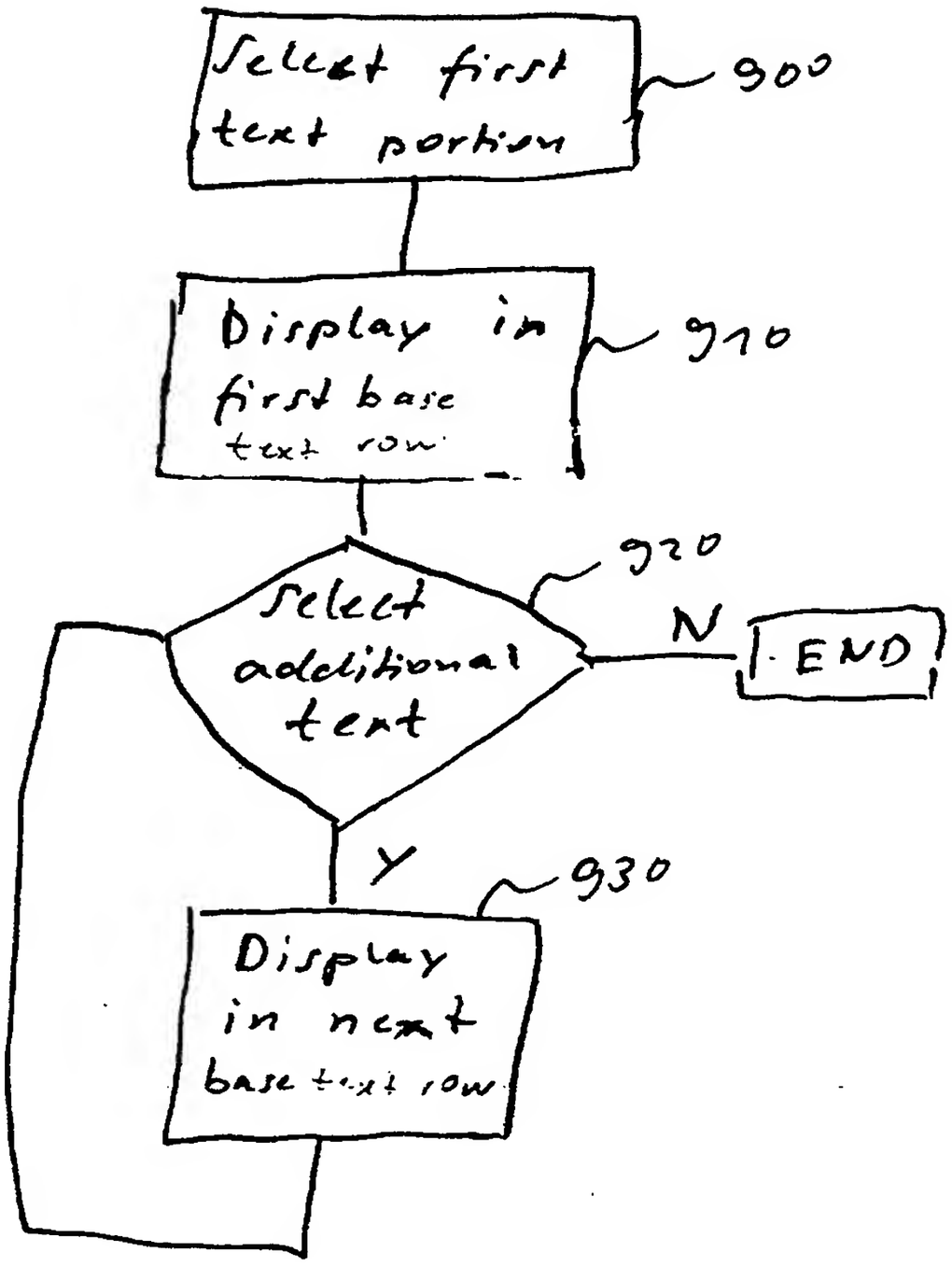

Select first text portion
900
Display in first base text row
910
Select additional text
920
N
END
Y
930
Display in next base text row

Fig. 9

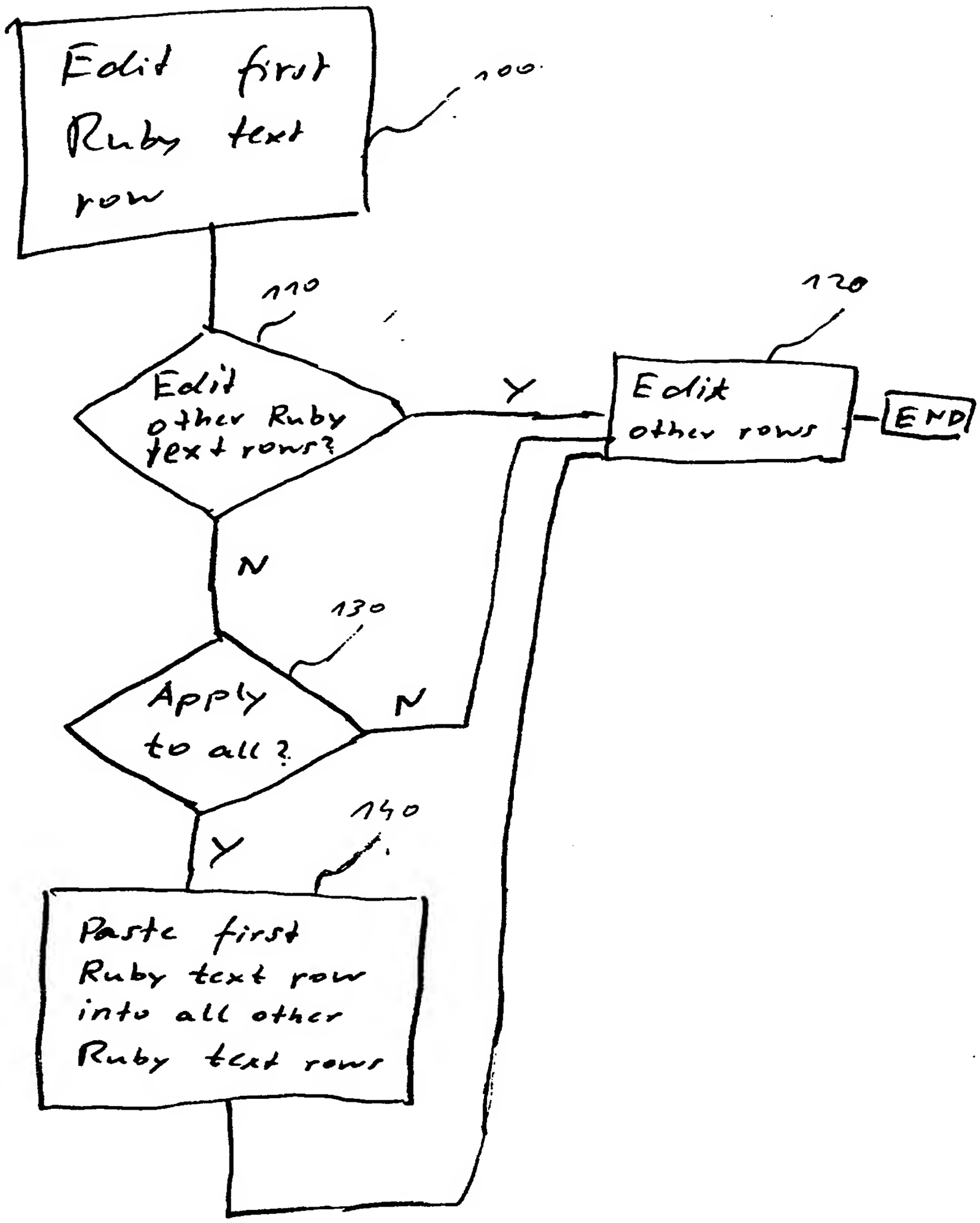

Edit first Ruby text row
100
110
Edit other Ruby text rows?
Y
120
Edit other rows
END
N
130
Apply to all?
N
Y
140
Paste first Ruby text row into all other Ruby text rows

Fig. 10

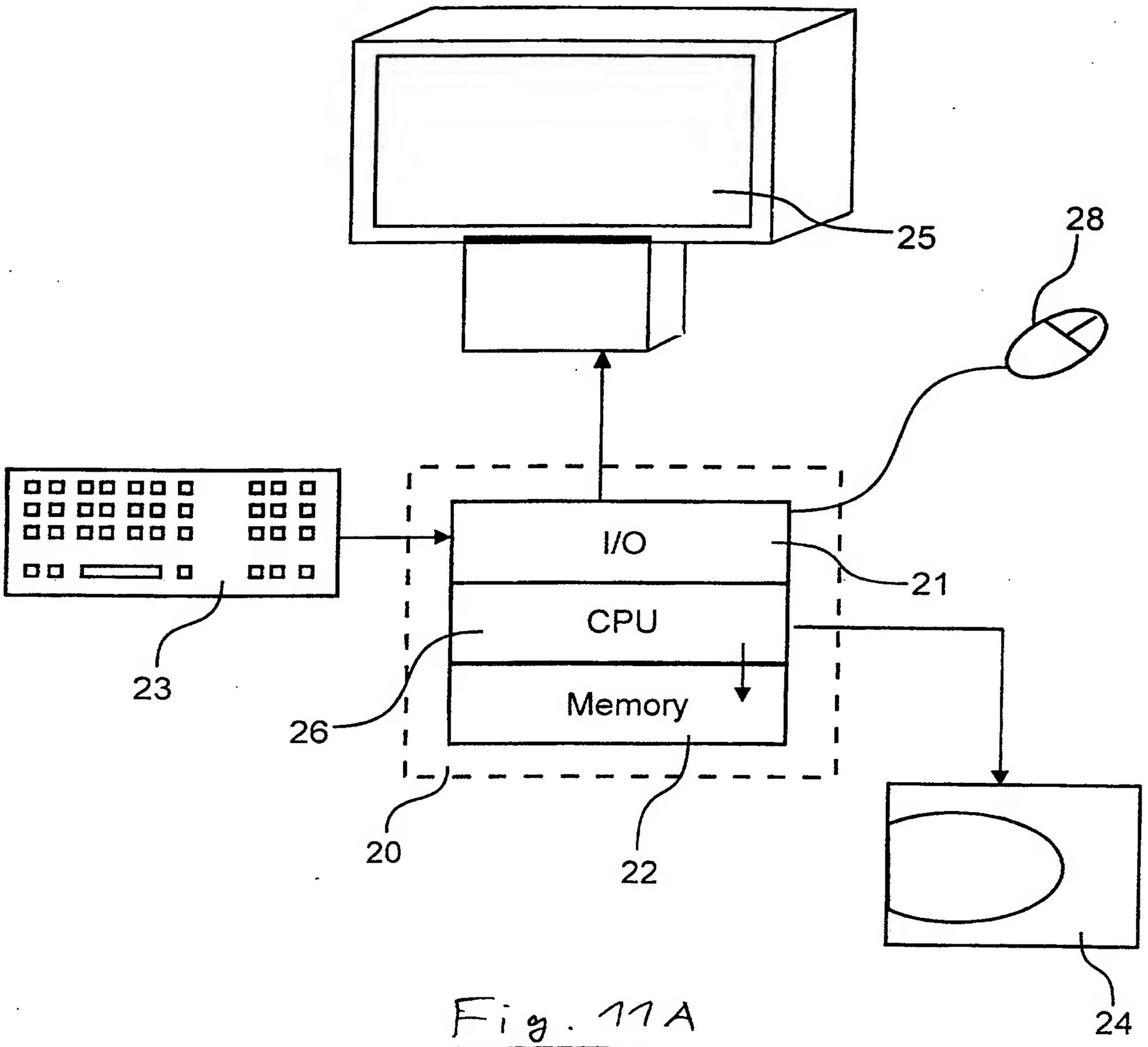


Fig. 11A

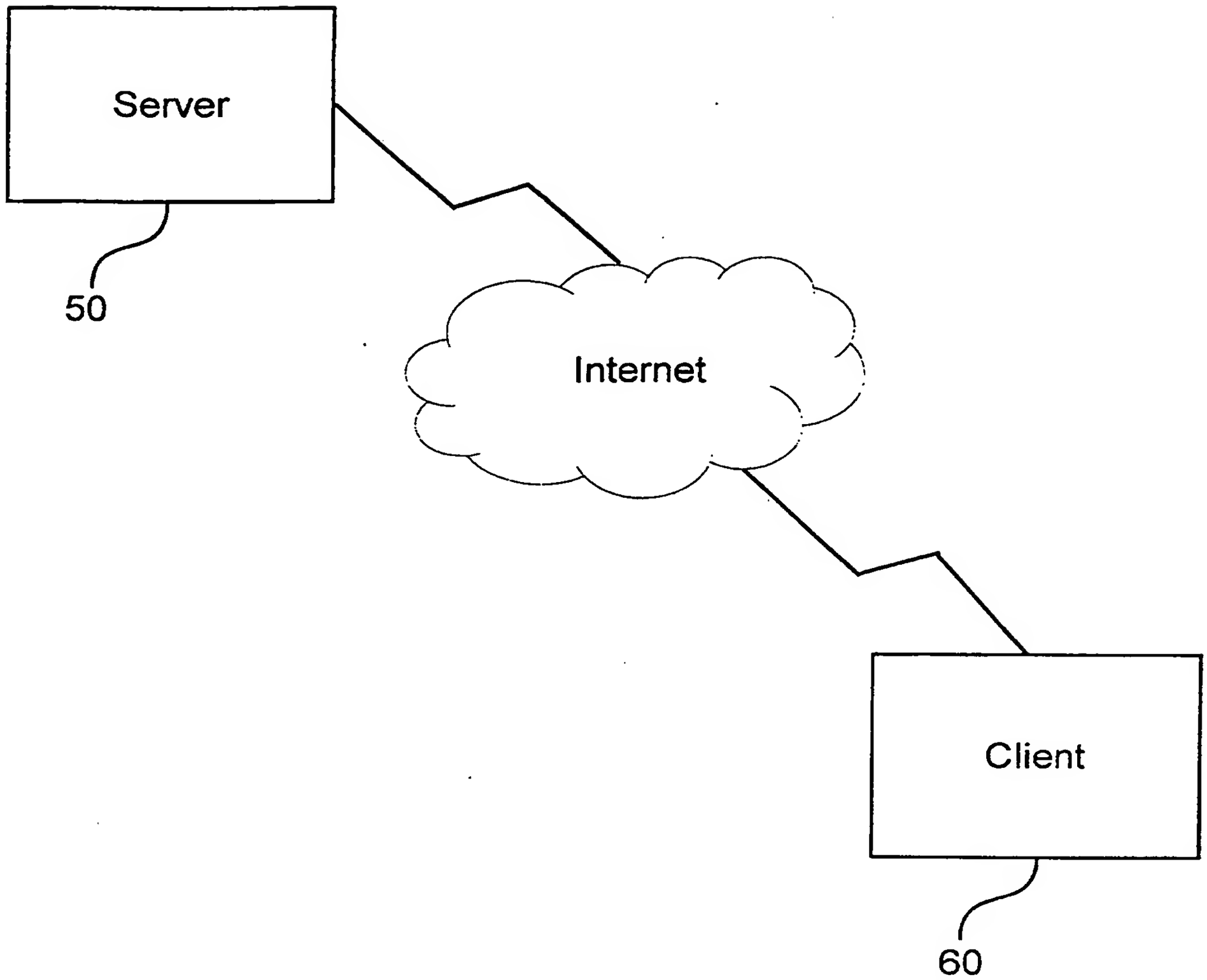


Fig. 11B

01 01 380.2

Appendix 1

W3C®

Ruby Annotation

W3C Working Draft 17 December 1999

This version:
http://www.w3.org/TR/1999/WD-ruby-19991217
(ZIP archive)
Latest version:
http://www.w3.org/TR/ruby
Previous version:
http://www.w3.org/TR/1999/WD-ruby-19990924

Editors:
Marcin Sawicki (until 10 October, 1999)
Michel Suignard, Microsoft
Masayasu Ishikawa (石川 雅康), W3C
Martin Dürst, W3C
(See Acknowledgements for additional contributors)

Abstract

"Ruby" are short runs of text alongside the base text, typically used in East Asian documents to indicate pronunciation or to provide a short annotation. This specification defines markup for ruby. The specification is written so that this markup for ruby can be included as a module of XHTML 1.1 [XHTML11].

Status of This Document

The W3C Internationalization Working Group (I18N WG) (members only), with this 17 December 1999 Last Call Working Draft, invites comment on this specification. The Last Call period begins 17 December 1999 and ends 14 January 2000.

This is a W3C Working Draft for review by W3C members and other interested parties. It is a draft document and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use W3C Working Drafts as reference material or to cite them as other than "work in progress". The W3C will not allow early implementation to constrain its ability to make changes to this specification prior to final release. A list of current W3C Recommendations and other technical documents can be found at http://www.w3.org/TR.

After last call comments have been addressed, the Working Group expects to advance this specification to Candidate Recommendation, and then to Proposed Recommendation together with XHTML 1.1, into which it will be included by reference. While the actual markup structure will not be changed at that point in the process, the I18N WG and the editors will make the necessary technical adjustments in notation if such adjustments become necessary as a consequence of changes to XHTML 1.1.

Please send comments and questions regarding this document to i18n-editor@w3.org (archived for W3C members). Comments in languages other than English, in particular Japanese, are also welcome. Public discussion on this specification may take place on the mailing list www-international@w3.org (archive).

Due to its subject matter, and to make the examples more realistic, this document includes examples using a wide range of characters. Not all user agents may be able to display all characters, changing the configuration can improve the situation. Also, great care has been taken to serve this document in various character encodings to cover a wide range of user agents and configurations.

Contents

---

1. Introduction

This section is *informative*.

1.1 What is ruby?

"Ruby" is the commonly used name for a run of text that appears in the immediate vicinity of another run of text, referred to as the "base". Ruby serve as a pronunciation guide or a short annotation associated with the base text. Ruby are used frequently in Japan in most kinds of publications, such as books and magazines, but also in China, especially in schoolbooks. Figure 1.1.1 shows an example.

にほんご←*ruby text*
日本語←*ruby base*

Figure 1.1.1: Ruby giving the pronunciation of the base characters.

East Asian typography has developed various elements that do not appear in western typography. Most of

these can be addressed appropriately with style sheet languages such as CSS or XSL. Ruby, however, require markup in order to define the association between the base text and the ruby text.

This specification defines such markup, designed to be usable with XHTML, in order to make ruby available on the Web without using special workarounds or graphics. This section gives some background on ruby. Section 1.2 gives an overview of the markup for ruby. Section 2 contains the formal definition of ruby markup in the context of the XHTML Modularization framework [XHTMLMOD].

The remainder of this section gives some background on the structure and use of ruby. Because presentation is part of this background, it is discussed here. However, it should be noted that this document does not specify any mechanisms for presentation/styling of ruby; this is part of the respective stylesheet languages.

The font size of ruby text is normally about half the font size of the base text (see Figure 1.1.1). The name "ruby" in fact originated from the name of the 5.5pt font size in British printing, which is about half the 10pt font size commonly used for normal text.

There are several positions where the ruby text can appear relative to its base. For horizontal layout, where text appears along horizontal lines, ruby are most frequently placed above the base text (see Figure 1.1.1), i.e. before the line containing the base text. Sometimes, especially in educational texts, ruby may appear below, i.e. after, the base text. In Chinese, it is rather common that Pinyin ruby appear after the base text.

日本語←*ruby base*
ni-hon-go ←*ruby text*

Figure 1.1.2: Ruby (in Latin letters) below the base text (in Japanese)

In typical vertical layout, where text appears along vertical lines starting on the right, ruby appear on the right side of (i.e. again before) the vertical line if they appear above in horizontal layout. The writing direction of the ruby text is the same as that of its base, that is vertical if the base is vertical, and horizontal if the base is horizontal.

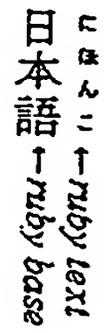


Figure 1.1.3: Ruby in vertical text (before/to the right)

Ruby text appears on the left side of the base in vertical layout if it appears below it in horizontal layout.

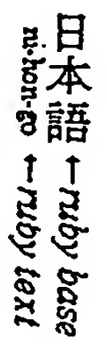


Figure 1.1.4: Ruby in vertical text (after/to the left).

Because East Asian text may be rendered vertically as well as horizontally, the terms "before" and "after" are used in this document rather than "above" and "below" or "right side" and "left side". In this specification, the

term "before" means "above" in horizontal layout / "right side" in vertical (in this context, top-to-bottom, right-to-left) layout, and the term "after" means "below" in horizontal layout / "left side" in vertical layout.

Ruby before the base text are often used to indicate pronunciation; ruby after the base text are often used to indicate meaning. In this and other cases, it can happen that ruby appear on both sides of the base text.

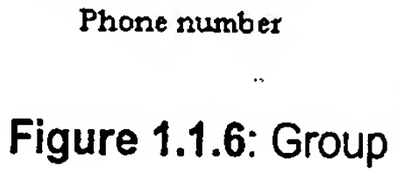


Figure 1.1.5: Ruby applied before and after a line of horizontal Japanese text

In some cases, it is desirable to give more details about which parts of the ruby base and which parts of the ruby text are associated together. Such a structure is called *group ruby*. This can be used for finetuning of the display or for other operations. Using fine-grained associations, and in particular showing the association with each single character of the base text, is mainly used in educational texts and other cases where the exact association is important or potentially unknown.

More coarse-grained association is used when the actual details of the association on a lower level are assumed to be known to the reader anyway, or unknown because the pronunciation or annotation only applies to the whole unit and cannot be split into pieces. Because in such cases longer spans of ruby text are set with the same spacing, better readability and more even layout may be achieved. For example, a person name can be decomposed into family name and given name. Or a kanji compound or phrase can be decomposed to show semantic subparts, as in the following example:

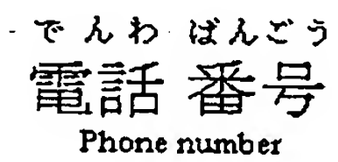


Figure 1.1.6: Group ruby with text after spanning the group

In the above example, the ruby text before the base text is made of two sequences: the hiragana sequence p"でんわ" (denwa) for "phone", and the hiragana sequence "ばんごう" (bango) for "number"; the ruby text after the base text is a single English sequence: "Phone number". Note that the space in the base text in Figure 1.1.6 and similar figures was added to make the structure of the examples clearer; in general, no space is used between ideographs in the base text.

In the following example, the ruby text after, "University", relates to the second base sequence while the ruby texts before, "けいおうぎじゅく" and "だいがく" (keiou gijyuku daigaku; Keio University in hiragana) refer to the two base sequences.

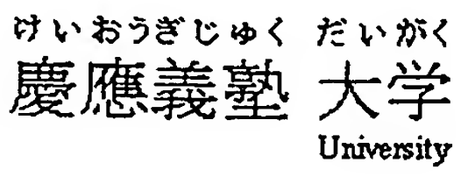


Figure 1.1.7: Group ruby with text after only spanning the second part

Details of ruby formating in a Japanese print context can be found in JIS-X-4051 [JIS].

In traditional Chinese, "Bopomofo" ruby can appear along the right side of the ruby base, even in horizontal layout.

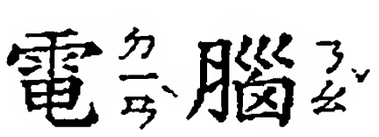


Figure 1.1.8: "Bopomofo" ruby in traditional Chinese (ruby text shown in blue for clarity) in horizontal layout

Note that Bopomofo tone marks (in the above example shown in red for clarity) appear in a separate column (along the right side of the Bopomofo ruby) and therefore might be seen as "ruby on ruby". However, they are encoded as combining characters that are simply part of the ruby text.

Introducing ruby to the Web leads to some phenomena and problems that are not present in traditional typography where the term "ruby" is taken from. The term "ruby" in Japanese is only used for text alongside the base text, for example as shown in the various figures above. However, once structural markup for ruby is defined as done in this specification, there is no guarantee that the associations defined by this markup will always be rendered alongside the base text. There is and will be a very wide variety of current and future output devices for documents marked up with XHTML. The following are possible scenarios and reasons for different rendering:

- Some user agents might not understand ruby markup, and may not be able to render ruby appropriately. In this case, it is preferable to nevertheless render the contents of ruby elements, in order not to lose information. The most acceptable fallback will be to place the ruby text immediately after the base text, and to enclose the ruby text in parentheses to reduce the potential for confusion (see Figure 1.1.9 below). However, it should be noted that this is only a fallback, and text in parentheses in Japanese typography is never called "ruby". Also, when ruby are rendered alongside the base text, they are not enclosed in parentheses.
- On display devices with low resolution, displaying ruby at half the point size of the base text may not be feasible. Again, a fallback with parentheses may be used.
- For educational purposes, it may be interesting to hide the ruby text and make it available as a pop-up. This is impossible on paper, but easily possible on a dynamic display device.
- On non-visual user agents such as voice browsers and braille user agents, only sequential rendering is possible. Depending on the function of the annotations, both the base text and the annotations, or only the annotations, may be rendered.

日本語(にほんご)
ruby base ruby text

Figure 1.1.9: Fallback for the example of Figure 1.1.1 using parentheses

Using parentheses for the fallback may lead to confusion between runs of text intended to be the ruby text and others that happen to be enclosed within parentheses. The document or style sheet author should be aware of the potential for that confusion and is advised to choose an unambiguous delimiter for the fallback, if this is a concern.

1.2 Ruby markup

This section gives an overview of the markup for ruby defined in this specification. A formal definition can be found in Section 2. The markup is written in XML [XML].

The core of the markup is the `ruby` element. The `ruby` element encloses all the text and markup necessary for defining the association between the base text and the ruby texts.

Note. *The name of this enclosing element, "`ruby`", should be interpreted to mean that what follows is associating ruby with base text. It must not be misunderstood to mean that everything inside, including the base text, is ruby. The name of the enclosing element was choosen to compactly and clearly identify the function of the markup construct; the names for the other elements were choosen to keep the overall length short.*

The `ruby` element serves as a container for one of the following:

- a combination of `rb`, `rt` and possibly `rp` elements (for simple cases)
- a combination of a single `rbc` and one or two `rtc` elements (for more complicated cases, such as group ruby).

In the following, these two cases are discussed in more detail.

1.2.1 Simple Ruby markup

For the simple case, the `rb` element contains the base text, the `rt` element contains the ruby text, and the optional `rp` elements contain the parenthesis characters used in the fallback case. `rb` stands for "ruby base", `rt` for "ruby text", and `rp` for "ruby parenthesis". This allows a simple association between one base text and one ruby text, and is sufficient for most cases.

For example, the following simple ruby:

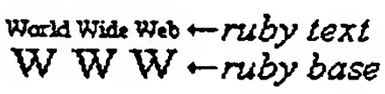


Figure 1.2.1.1: Ruby applied to English

can be represented as follows:

```
<ruby>
  <rb>WWW</rb>
  <rp>(</rp><rt>World Wide Web</rt><rp>)</rp>
</ruby>
```

Figure 1.2.1.2: Example of simple ruby markup including `rp` elements for fallbacks

The `rp` elements and the parentheses inside them are provided for fallback only. Some user agents, which ignore unknown elements but render their contents, will render the above markup as **"WWW (World Wide Web)"**. The `rp` element identifies the parentheses (or whatever else that may be used in their place) to user agents that know about the markup defined in this specification so that the parentheses can be removed. If the author is not concerned about fallbacks for user agents that neither know about ruby markup nor support CSS2 [CSS2] or XSL [XSL] style sheets, then the `rp` elements are not needed:

```
<ruby>
  <rb>WWW</rb>
  <rt>World Wide Web</rt>
</ruby>
```

Figure 1.2.1.3: Example of simple ruby markup without `rp` elements for fallbacks

In [CSS2], if necessary, the parentheses can be generated using the 'content' property ([CSS2], section 12.2) with the :before and :after pseudo-elements ([CSS2], section 12.1), as for example in the following style declaration:

```
rt:before { content: "(" }
rt:after { content: ")" }
```

Figure 1.2.1.4: CSS2 style sheets to generate parentheses around `rt` element

In the above example, parentheses will be automatically generated around the `rt` element. It is assumed

that the above style rules are used together with style rules that position the ruby text inline. Generation of parentheses is straightforward with XSLT [XSLT].

1.2.2 Group Ruby markup

For more complicated cases of associations between a base text and some ruby texts, a combination of `rbc` and `rtc` elements is used. This includes associating more than one ruby texts with the same base text (typically rendered on both sides of the base text) and fine-grained associations of the base text and the ruby text (group ruby). The `ruby` element contains one `rbc` element followed by one or two `rtc` elements. The `rbc` element contains `rb` elements, the `rtc` element contains `rt` elements. Several `rb` elements inside an `rbc` element, combined with several `rt` elements inside an `rtc` element, are used for group ruby. The `rt` element may use the `rbspan` attribute to indicate that a single `rt` element spans (is associated with) multiple `rb` elements. This is similar to the `colspan` attribute of the `th/td` elements in tables. The `rbc` stands for "ruby base component", and the `rtc` for "ruby text component".

An example of group ruby is shown in the following figure:

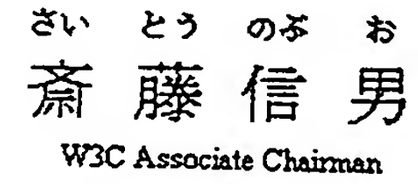


Figure 1.2.2.1: Group ruby with mixed before and after ruby text

can be represented as follows:

```
<ruby xml:lang="ja" class="pronunciation annotation">
  <rbc>
    <rb>斎</rb>
    <rb>藤</rb>
    <rb>信</rb>
    <rb>男</rb>
  </rbc>
  <rtc class="pronunciation">
    <rt>さい</rt>
    <rt>とう</rt>
    <rt>のぶ</rt>
    <rt>お</rt>
  </rtc>
  <rtc class="annotation">
    <rt rbspan="4" xml:lang="en">W3C Associate Chairman</rt>
  </rtc>
</ruby>
```

Figure 1.2.2.2: Ruby markup to achieve both before and after ruby texts on the same base.

In this case, the `rtc` element with the class "`pronunciation`" should be associated with the style information that places the ruby texts before the ruby bases, and the `rtc` element with the class "`annotation`" should be associated with the style information that places the ruby texts after the ruby bases.

The `rp` element is not available in this representation. This has two reasons. First, the `rp` element is for fallback only, and it was considered that this is much more important for the more frequent simple case. Second, for the more complex cases, it is in many cases very difficult to come up with a reasonable fallback display, and constructing markup for such cases can be even more difficult if not impossible.

1.2.3 Notes on styling ruby

This specification only defines ruby markup. Formatting properties for styling ruby are under development for CSS and XSL. See "*International Layout*" [I18N-FORMAT] (*work in progress*) for more details.

Note that for non-visual rendering such as speech synthesis, rendering both the base text and the ruby text can be annoying. This is in particular the case if the ruby represent a pronunciation. In this case, a speech synthesizer may either be able to correctly pronounce the base text, in which case the same text is spoken twice, or it may not know the correct pronounciation of the text and make up a pronounciation, in which case the result may be quite confusing.

As an example, in the case of Figure 1.2.2.2, the ruby bases "斎", "藤", "信", "男" are less useful than the ruby texts "さい" (sai), "とう" (tou), "のぶ" (nobu), "お" (o) for non-visual rendering, and it does not make sense to render both the ruby bases and the ruby texts. Because in this case, the ruby texts are used to represent pronunciation, so it is straightforward to use them for non-visual rendering. In such cases, something like the following style information may help.

```
@media aural {
  ruby[class~="pronunciation"] rb { speak: none }
}
```

Figure 1.2.3.1: CSS2 style sheet to suppress aural rendering of the ruby base

The above style sheet will suppress aural rendering of the ruby base when the `rb` element is a descendant element of the `ruby` element with the class "`pronunciation`". See [CSS2] for more details.

Note that ruby used as a pronunciation guide may be different from the actual pronunciation even in cases where the script used for indicating the pronunciation at first glance seems perfectly phonetic. For example, Bopomofo is associated independently for each base character; context-dependent sound or tone changes are not reflected in ruby. Similarly, in Japanese, spelling irregularities can occur, such as using "は" (ha) for the topic suffix pronounced "わ" (wa). For such cases, authors may want to supply two variants, distinguished by the value of the `class` attribute, or may rely on the aural rendering system being able to handle such cases correctly.

It is also important to note that not all ruby are pronunciations. Authors should distinguish ruby used for different purposes by using the `class` attribute, as done above by assuming `class="pronunciation"` for ruby used to indicate pronunciation. Also, it should be noted that somebody listening to aural rendering may be interested in accessing the skipped base text to check the characters used.

---

2. Formal definition of ruby elements

This section is *normative*.

This section contains the formal syntax definition and the specification of the functionality of the ruby elements. Some familiarity with the XHTML Modularization framework, in particular, the "*Modularization of XHTML*" [XHTMLMOD] and the "*Building XHTML Modules*" [BUIDLING] specifications is assumed.

2.1 Abstract definition of ruby elements

The following is the abstract definition of ruby elements, which is consistent with the XHTML Modularization framework [XHTMLMOD]. Further definitions of XHTML abstract modules can be found in [XHTMLMOD].

Elements	Attributes	Minimal Content Model
ruby	Common	(rb, rp?, rt, rp?)
rbc	Common	rb+
rtc	Common	rt+
rb	Common	(PCDATA | Inline - ruby)*
rt	Common, rbspan (CDATA)	(PCDATA | Inline - ruby)*
rp	Common	PCDATA*

The ruby content model for XHTML 1.1 [XHTML11] is as follows:

```
((rb, rp?, rt, rp?) | (rbc, rtc, rtc?))
```

This is the maximal content model for the ruby element.

An implementation of this abstract definition as an XHTML DTD module can be found in Appendix A. An XML Schema [XMLSchema] implementation will be provided when feasible.

2.2 The ruby element

The ruby element is an inline (or text-level) element that serves as the container for either the rb, rt and optional rp elements or the rbc and rtc elements. It provides the structural association between the ruby base elements and their ruby text elements.

The ruby element has common attributes only, such as id, class or xml:lang.

In this simplest example, the ruby text "aaa" is associated with the base "AA":

```
<ruby><rb>AA</rb><rt>aaa</rt></ruby>
```

Figure 2.2.1: Simple usage of the ruby element

2.3 The rbc element

The rbc (ruby base component) element is the container for rb elements. This element is only used for group ruby. Only one rbc element may appear inside a ruby element. Examples of using the rbc element are shown in Figure 1.2.2.2 and Figure 2.4.1.

The rbc element has common attributes only.

2.4 The rtc element

The rtc (ruby text component) element is the container for rt elements. This element is only used for group ruby. One or two rtc elements may appear inside a ruby element to associate ruby texts with a single ruby base, represented by an rbc element. More than two rtc elements MUST not appear inside a ruby element.

The rtc element has common attributes only.

For example, the following markup associates two ruby texts with the same ruby base:

```
<ruby>
  <rbc>
    <rb>KANJI</rb>
  </rbc>
  <rtc class="before">
    <rt>kana-before</rt>
  </rtc>
  <rtc class="after">
    <rt>kana-after</rt>
  </rtc>
</ruby>
```

Figure 2.4.1: Ruby markup to associate two ruby texts with the same base.

Supposing that the class "`before`" is used to apply style information to place the ruby text before the ruby base and the class "`after`" is used to apply style information to place the ruby text after the ruby base, in horizontal text the markup above would be rendered like this:

```
kana-before
   KANJI
kana-after
```

Figure 2.4.2: Horizontal rendering of two ruby texts associated with a single ruby base.

Markup for a more complex example of using two ruby texts with the same ruby base is shown in Figure 1.2.2.2; the rendering of this markup is shown in Figure 1.2.2.1.

Note. *Although the rendering of the ruby texts should be controlled by style sheets, in case no style information is provided by the author or the user, it is recommended that visual user agents should place the ruby text before the ruby base when only one ruby text is used. This is also the case for simple ruby. When there are two ruby texts, the first ruby text should be placed before the ruby base, and the second ruby text should be placed after the ruby base. A sample user agent default style sheet which describes this formatting will be provided by [I18N-FORMAT] or its successor document.*

For non-visual rendering, in the absence of style sheet information, it is recommended that both the ruby base and the ruby text(s) should be rendered, with an indication (e.g. different voice, different pitch, ...) of the status of each.

2.5 The `rb` element

The `rb` element is the container for the text of the ruby base. For simple ruby, only one `rb` element may appear. An example of such simple ruby is shown in Figure 2.2.1. For group ruby, multiple `rb` elements may appear inside an `rbc` element, and each `rb` element may be associated with the relevant `rt` element, for fine-grained association. An example of such group ruby is shown in Figure 1.2.2.2.

The `rb` element may contain inline elements or character data as its content, but the `ruby` element shall not appear as its descendant element.

The `rb` element has common attributes only.

2.6 The `rt` element

The `rt` element is the container for the ruby text. For simple ruby, only one `rt` element may appear. An

example of such simple ruby is shown in Figure 2.2.1. For group ruby, multiple `rt` elements may appear inside an `rtc` element, and each `rt` element contains the ruby text for the relevant ruby base, represented by the `rb` element. An example of such fine-grained association between the ruby bases and the ruby texts is shown in Figure 1.2.2.2, and that would be rendered like Figure 1.2.2.1.

The `rt` element may contain inline elements or character data as its content, but the `ruby` element shall not appear as its descendant element.

The `rt` element has common attributes and the `rbspan` attribute. In group ruby, the `rbspan` attribute allows an `rt` element to span multiple `rb` elements. The value shall be an integer value greater than zero ("0"). The default value of this attribute is one ("1").

In an example shown in Figure 1.2.2.2, the secondary ruby text spans four `rb` elements, and it will be rendered as in Figures 1.2.2.1. The `rbspan` attribute should not be used in simple ruby, and user agents should ignore the `rbspan` attribute when it appears in simple ruby.

2.7 The `rp` element

The `rp` element is intended to contain parenthesis characters in simple ruby. Parentheses are necessary to provide an acceptable fallback. The `rp` element is necessary especially for user agents that are unable to render ruby text alongside the ruby base. In that way, ruby will only degrade to be rendered inline and enclosed in parentheses. This is the least inappropriate rendering under the condition that only inline rendering is available. The `rp` element cannot be used in group ruby.

The `rp` element has common attributes only.

Consider the following markup:

```
<ruby>
  <rb>A</rb>
  <rp>(</rp><rt>aaa</rt><rp>)</rp>
</ruby>
```

Figure 2.7.1: Ruby markup using `rp` elements

A user agent that supports ruby will render it as:

```
aaa
 A
```

Figure 2.7.2: Ruby rendered by a supporting user agent (the parentheses have been removed)

A user agent that is unable to render ruby alongside the ruby base, or does not support ruby markup, will render it as:

```
A (aaa)
```

Figure 2.7.3: Ruby rendered by a non-supporting user agent (the parentheses are visible)

Appendices

A. Ruby module for XHTML

This appendix is *informative*.

The following is a link to the Ruby DTD module that will be part of the XHTML 1.1 DTD modules [XHTML11].

- XHTML 1.1 Ruby Module

B. Notes on backwards compatibility

This appendix is *informative*.

For historical reason, some authoring tools might generate ruby markup without the start and end tags of the `rb` element, like:

```
<ruby>
  A
  <rp>(</rp><rt>aaa</rt><rp>)</rp>
</ruby>
```

rather than the following:

```
<ruby>
  <rb>A</rb>
  <rp>(</rp><rt>aaa</rt><rp>)</rp>
</ruby>
```

The former markup is not conforming to this specification, but user agents that care about compatibility with documents generated by such authoring tools may treat the former markup as if it were written like the latter.

C. Glossary

This appendix is *informative*.

Bopomofo
: 37 characters and 4 tone marks used as phonetics in Chinese, especially standard Mandarin.

Group ruby
: A set of ruby that uses fine-grained associations between the ruby bases and the ruby texts.

Hiragana
: Subset of the Japanese writing system consisting of phonetic characters to represent Japanese words.

Kana
: Syllabic subset of the Japanese system of writing that can be used exclusively for writing foreign words or in combination with kanji.

Kanji
: Subset of the Japanese writing system that utilizes ideographic characters borrowed or adapted from Chinese writing.

Ruby base
: Run of text that has a ruby text associated with it.

Ruby text
: Run of text that appears in the immediate vicinity of another run of text (called "ruby base") and serves as an annotation or a pronunciation guide associated with the base.

This section is *informative*.

The model presented in this specification was originally inspired by the work done by Martin Dürst [DUR97].

This specification would also not have been possible without the help from:

Murray Altheim, Mark Davis, Laurie Anna Edlund, Arye Gittelma, Hideki Hiura (樋浦　秀樹), Koji Ishii, Rick Jelliffe, Eric LeVine, Chris Lilley, Charles McCathieNevile, Shigeki Moro (師　茂樹), Chris Pratley, Nobuo Saito (斎藤　信男), Rahul Sonnad, Takao Suzuki (鈴木　孝雄), Chris Thrasher, Chris Wilson, Masafumi Yabe (家辺　勝文).

References

This section is *informative*.

[BUILDING]
"Building XHTML™ Modules", W3C Working Draft
M. Altheim, S. P. McCarron, eds., 10 September 1999
Available at: http://www.w3.org/TR/1999/WD-xhtml-building-19990910
The latest version is available at: http://www.w3.org/TR/xhtml-building

[CSS2]
"Cascading Style Sheets, level 2 (CSS2) Specification", W3C Recommendation
B. Bos, H. W. Lie, C. Lilley and I. Jacobs, eds., 12 May 1998
Available at: http://www.w3.org/TR/1998/REC-CSS2-19980512
The latest version is available at: http://www.w3.org/TR/REC-CSS2

[DUR97]
"Ruby in the Hypertext Markup Language", Internet Draft
Martin Dürst, 28 February 1997, *expired*
Available at: http://www.w3.org/International/draft-duerst-ruby-01

[I18N-FORMAT]
"International Layout", W3C Working Draft
M. Sawicki, ed., 10 September 1999
Available at: http://www.w3.org/TR/1999/WD-i18n-format-19990910
The latest version is available at: http://www.w3.org/TR/i18n-format

[JIS]
"Line composition rules for Japanese documents"
JIS X 4051-1995, Japanese Standards Association, 1995 (in Japanese)

[XHTML11]
"XHTML™ 1.1 - Module-based XHTML", W3C Working Draft
M. Altheim, S. McCarron, Eds., 10 September 1999
Available at: http://www.w3.org/TR/199/WD-xhtml11-19990910
The latest version is available at: http://www.w3.org/TR/xhtml11

[XHTMLMOD]
"Modularization of XHTML™", W3C Working Draft
M. Altheim *et. al.*, eds., 10 September 1999
Available at: http://www.w3.org/TR/1999/WD-xhtml-modularization-19990910
The latest version is available at: http://www.w3.org/TR/xhtml-modularization

[XML]
"Extensible Markup Language (XML) 1.0 Specification", W3C Recommendation
T. Bray, J. Paoli, C. M. Sperberg-McQueen, eds., 10 February 1998
Available at: http://www.w3.org/TR/1998/REC-xml-19980210

D. Changes from previous public Working Draft

This appendix is *informative*.

Section	Change
Abstract	• Rewritten substantially; made clear that this specification is defining a ruby module for XHTML 1.1.
Status of This Document	• Rewritten substantially; made clear that this document is a Last Call draft. • Added a note on the use of wide range of characters and character encodings, and provided a link to a list of available character encodings.
1. Introduction	• Changed above/below to before/after so that it makes sense regardless of layout flow. • Splitted section 1.2 into three subsections. • Added a note that in Chinese, Pinyin ruby commonly appears "after" the ruby base rather than "before". • Added an example of Bopomofo ruby, and added a note about tone marks in Bopomofo ruby. • Added a note on possible difference between ruby and the actual pronunciation. • Refined some wording.
2. Formal definition of ruby elements	• Changed the "minimal" content model of the `ruby` element to `(rb, rp?, rt, rp?)` • Changed the content model of the `ruby` element of XHTML 1.1 to limit the number of `rtc` elements to no more than two in group ruby. • Added note on visual/non-visual rendering of ruby text. • Refined each definition.
Appendix A. Ruby module for XHTML	• Rewritten Ruby module substantially for better modularity • Removed accompanying DTD modules, as those will be incorporated into XHTML 1.1.
Appendix B. Notes on backwards compatibility	• Removed an appendix on ruby usage in SGML, and added notes on backwards compatibility instead.
Acknowledgements	• Added additional contributors, and used kanji characters as well for their names when appropriate.
References	• Removed references to HTML 4.0 and XHTML 1.0. • Added references to XML Schema and XSLT specifications. • Added references to "this version" URIs in addition to "latest version" URIs.

Acknowledgements

The latest version is available at: http://www.w3.org/TR/REC-xml

[XMLSchema]

"XML Schema Part 1: Structures", W3C Working Draft
H. S. Thompson, D. Beech, M. Maloney, N. Mendelsohn, eds., 5 November 1999
Available at: http://www.w3.org/TR/1999/WD-xmlschema-1-19991105
The latest version is available at: http://www.w3.org/TR/xmlschema-1
See also "*XML Schema Part 2: Datatypes*", available at: http://www.w3.org/TR/xmlschema-2

[XSL]

"Extensible Style Language (XSL)", W3C Working Draft
S. Deach, ed., 21 April 1999
Available at: http://www.w3.org/TR/1999/WD-xsl-19990421
The latest version is available at: http://www.w3.org/TR/WD-xsl

[XSLT]

"XSL Transformations (XSLT) Version 1.0", W3C Recommenation
J. Clark, ed., 16 November 1999
Available at: http://www.w3.org/TR/1999/REC-xslt-19991116
The latest version is available at: http://www.w3.org/TR/xslt